VISUAL BASIC FOR APPLICATIONS (VBA) CODE PROFILER

USER & INSTALLATION GUIDE

Bandwood Pty Limited

ACN 052 052 346

Version Number: **1.19** (for software revision 2.4.4 or latter)

Date Issued: **22 August 2016**

TABLE OF CONTENTS

TABLE OF FIGURES

1 INTRODUCTION

1.1 PURPOSE

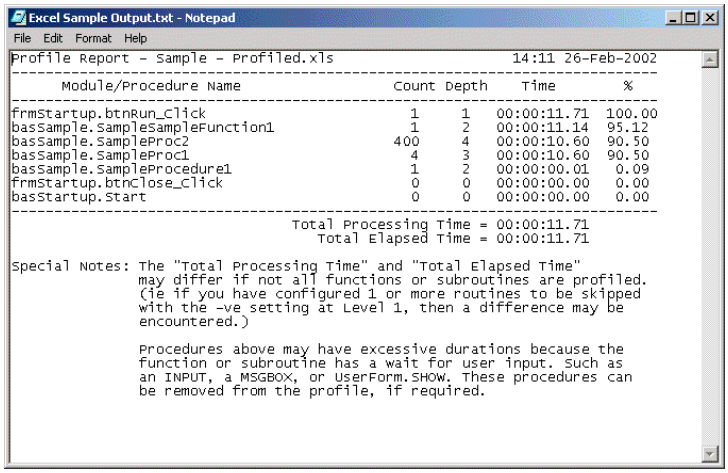
This document provides an Installation and User guide to the Visual Basic for Applications (VBA) Code Profiler (VBACP) system developed for analysing VBA code within the Microsoft® Excel® and Word® suite of products.

The VBACP software has been developed by Bandwood Pty Limited, Australia and has been developed for the Microsoft® Office Suite of software.

1.2 ABOUT VBACP

The VBA Code Profiler allows for the instrumentation and profiling of VBA code. Ultimately, a report similar to that depicted below will be produced, illustrating the number of times a routine is called, the time spent within each routine, and the level-of-nesting for each routine.

The VBA Code Profiler will operate on any Microsoft® VBA system and produces very similar formatted results. This will facilitate understanding the reports across the spectrum of VBA systems.



Excel Sample Output.txt - Notepad

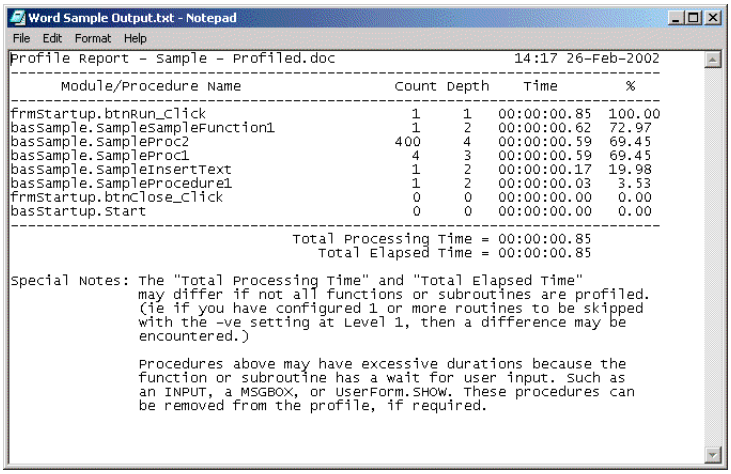
File Edit Format Help

```
Profile Report - Sample - Profiled.xls                    14:11 26-Feb-2002
-------------------------------------------------------------------------------
      Module/Procedure Name                  Count Depth    Time        %
-------------------------------------------------------------------------------
frmStartup.btnRun_Click                          1     1   00:00:11.71  100.00
basSample.SampleSampleFunction1                  1     2   00:00:11.14   95.12
basSample.SampleProc2                          400     4   00:00:10.60   90.50
basSample.SampleProc1                            4     3   00:00:10.60   90.50
basSample.SampleProcedure1                       1     2   00:00:00.01    0.09
frmStartup.btnClose_Click                        0     0   00:00:00.00    0.00
basStartup.Start                                 0     0   00:00:00.00    0.00
-------------------------------------------------------------------------------
                                  Total Processing Time = 00:00:11.71
                                     Total Elapsed Time = 00:00:11.71

Special Notes: The "Total Processing Time" and "Total Elapsed Time"
               may differ if not all functions or subroutines are profiled.
               (ie if you have configured 1 or more routines to be skipped
               with the -ve setting at Level 1, then a difference may be
               encountered.)

               Procedures above may have excessive durations because the
               function or subroutine has a wait for user input. Such as
               an INPUT, a MSGBOX, or UserForm.SHOW. These procedures can
               be removed from the profile, if required.
```

Figure 1 Excel Sample Output



Word Sample Output.txt - Notepad

File Edit Format Help

```
Profile Report - Sample - Profiled.doc                    14:17 26-Feb-2002
-------------------------------------------------------------------------------
      Module/Procedure Name                  Count Depth    Time        %
-------------------------------------------------------------------------------
frmStartup.btnRun_Click                          1     1   00:00:00.85  100.00
basSample.SampleSampleFunction1                  1     2   00:00:00.62   72.97
basSample.SampleProc2                          400     4   00:00:00.59   69.45
basSample.SampleProc1                            4     3   00:00:00.59   69.45
basSample.SampleInsertText                       1     2   00:00:00.17   19.98
basSample.SampleProcedure1                       1     2   00:00:00.03    3.53
frmStartup.btnClose_Click                        0     0   00:00:00.00    0.00
basStartup.Start                                 0     0   00:00:00.00    0.00
-------------------------------------------------------------------------------
                                  Total Processing Time = 00:00:00.85
                                     Total Elapsed Time = 00:00:00.85

Special Notes: The "Total Processing Time" and "Total Elapsed Time"
               may differ if not all functions or subroutines are profiled.
               (ie if you have configured 1 or more routines to be skipped
               with the -ve setting at Level 1, then a difference may be
               encountered.)

               Procedures above may have excessive durations because the
               function or subroutine has a wait for user input. Such as
               an INPUT, a MSGBOX, or UserForm.SHOW. These procedures can
               be removed from the profile, if required.
```

Figure 2 Word Sample Output

2 INSTALLATION

2.1 SYSTEM PRE-REQUISITES

The VBA Code Profiler (VBACP) system requires any of the Microsoft® Excel® VBA or Word® VBA systems from Microsoft® Office 97, 2000, XP, 2003, 2007, 2010, 2013, and 2016 suites of software.

The VBA Code Profiler (VBACP) system also supports Microsoft® Access® VBA for Microsoft® Office 2000, XP, 2003, 2007, 2010, 2013, and 2016 suites of software.

Note, it does not support Access 97.

To ensure the accuracy of the profiled times, the PC running the profiler MUST have been rebooted within the last 49 days. If your PC has been running longer than 49 days, then you may get incorrect timing information. This is necessary to ensure millisecond level timing.

2.2 SYSTEM INSTALLATION LOCATION & FILES

Run the ***'setup.exe'*** program on the media provided. You will be prompted for a directory to install (VBACP) into.

The complete installation will include the following files:

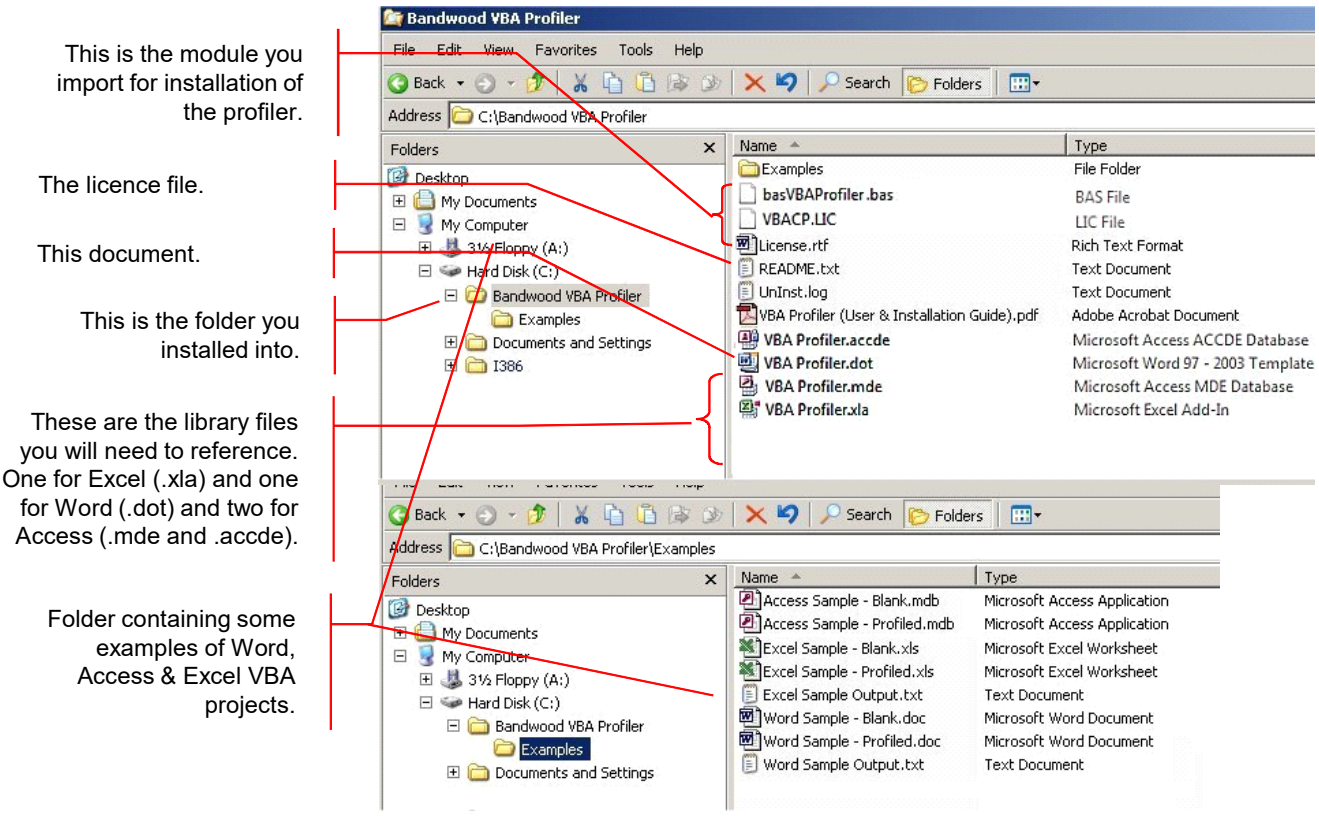


Figure 3 Installed Files

Important: when connecting in the appropriate library file, use ***VBA Profiler.xla*** for Microsoft® Excel®, ***VBA Profiler.dot*** for Microsoft® Word®, and ***VBA Profiler.mde*** for Microsoft® Access®.

3 USAGE GUIDE

To use the profiler, your code is required to be instrumented. This means, your code will be modified by the VBA Code Profiler (VBACP) to allow the measurements to be taken. Instrumentation and later Cleaning of your code can be performed manually, one project at a time, or via the installer that will instrument/clean multiple projects in a single pass.

SPECIAL NOTE:
Since the VBA Code Profiler modifies your code, it is HIGHLY RECOMMENDED that you backup your code before running the profiler.

3.1 CODE INSTRUMENTATION

Since the code instrumentation process can be sensitive to different coding styles (we have endeavoured to minimise this), it may have problems with your specific style. If problems are encountered, please send your code fragment to us at support@bandwood.com so we can continually improve the robustness of the profiler. Also, you may need to modify the affected code slightly to allow the profiler to function correctly.

3.1.1 VBA Security Warning

Under some installations of Microsoft® Office, notable 2000, XP and 2003, you may receive the following error message when trying to instrument your code, please follow the instructions below to rectify:

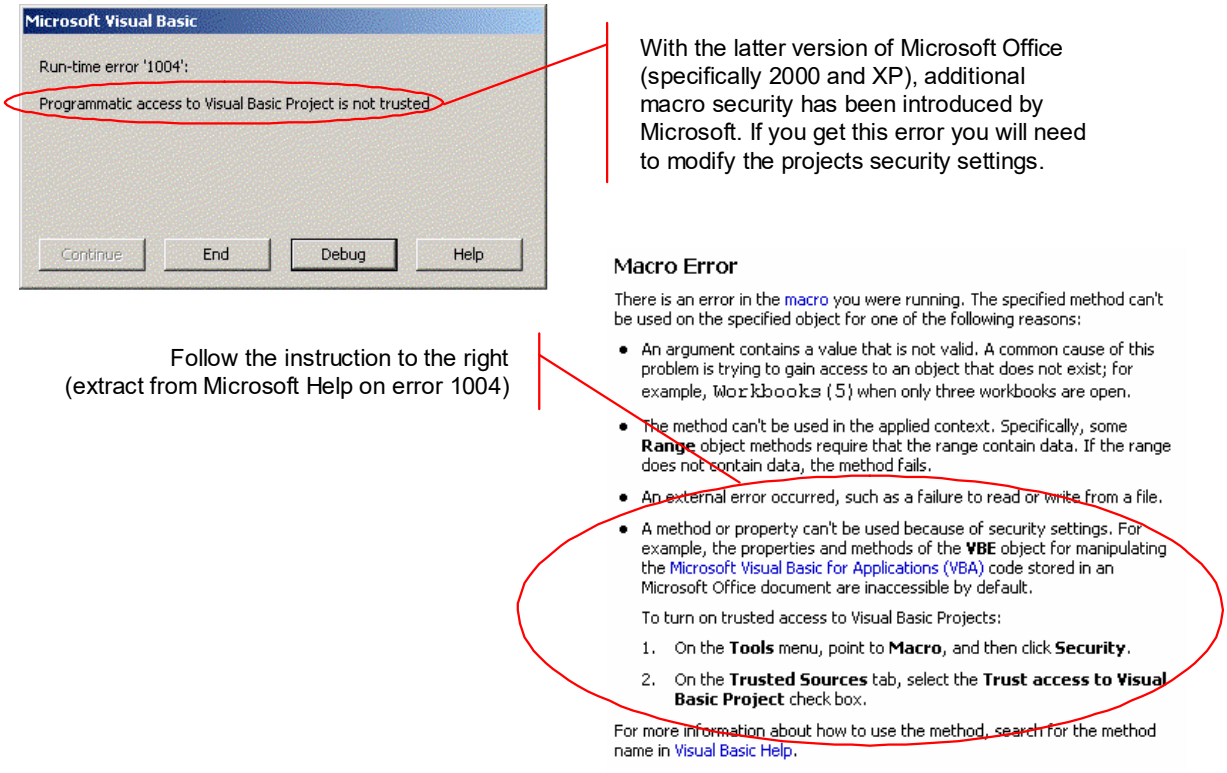


Figure 4 VBA Security Warning

3.1.2 Code Instrumentation

3.1.2.1 Step-By-Step Guide

To instrument your code, follow the steps illustrated below:

Step 1

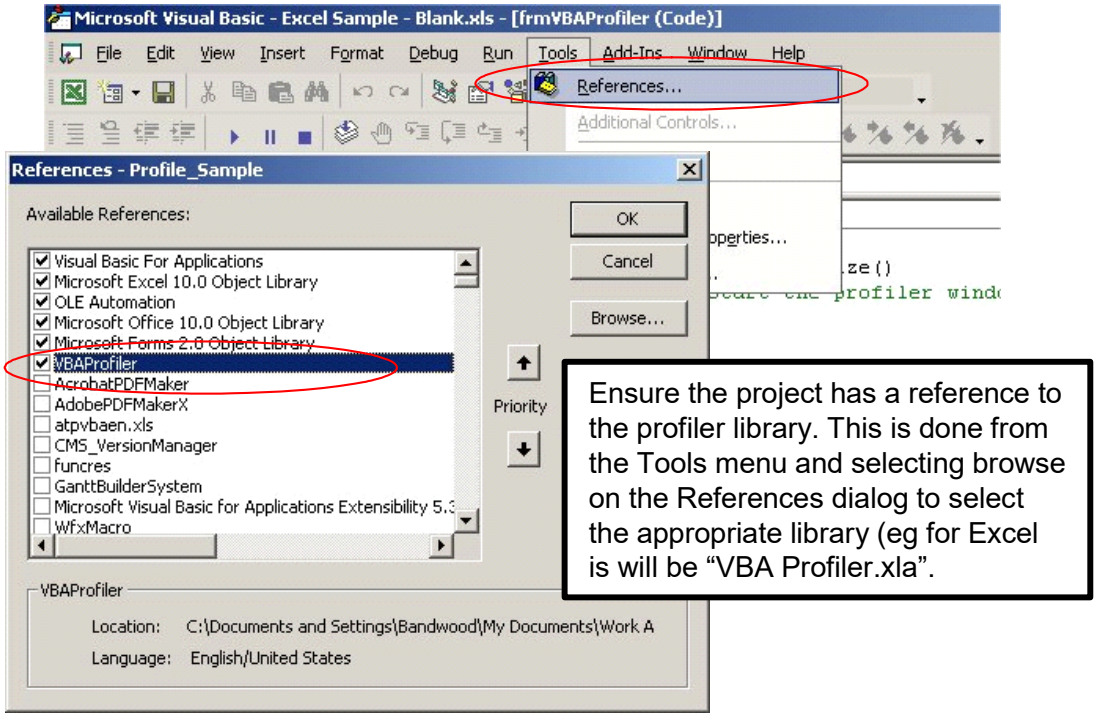


Figure 5 Instrumentation (Part 1)

Note, after you have selected the “Browse” button, change the file type from any DLL, OLB or TLB to one of the following;

- For Excel select .XLA
- For Word select .DOT
- For Access 2000, XP, 2003 select .MDE
- For Access 2007, 2010, 2013, 2016 select .ACCDE

then:

Step 2

In the Import File dialog, select the "***basVBAProfiler .bas***" file from the folder VBCP was installed into.

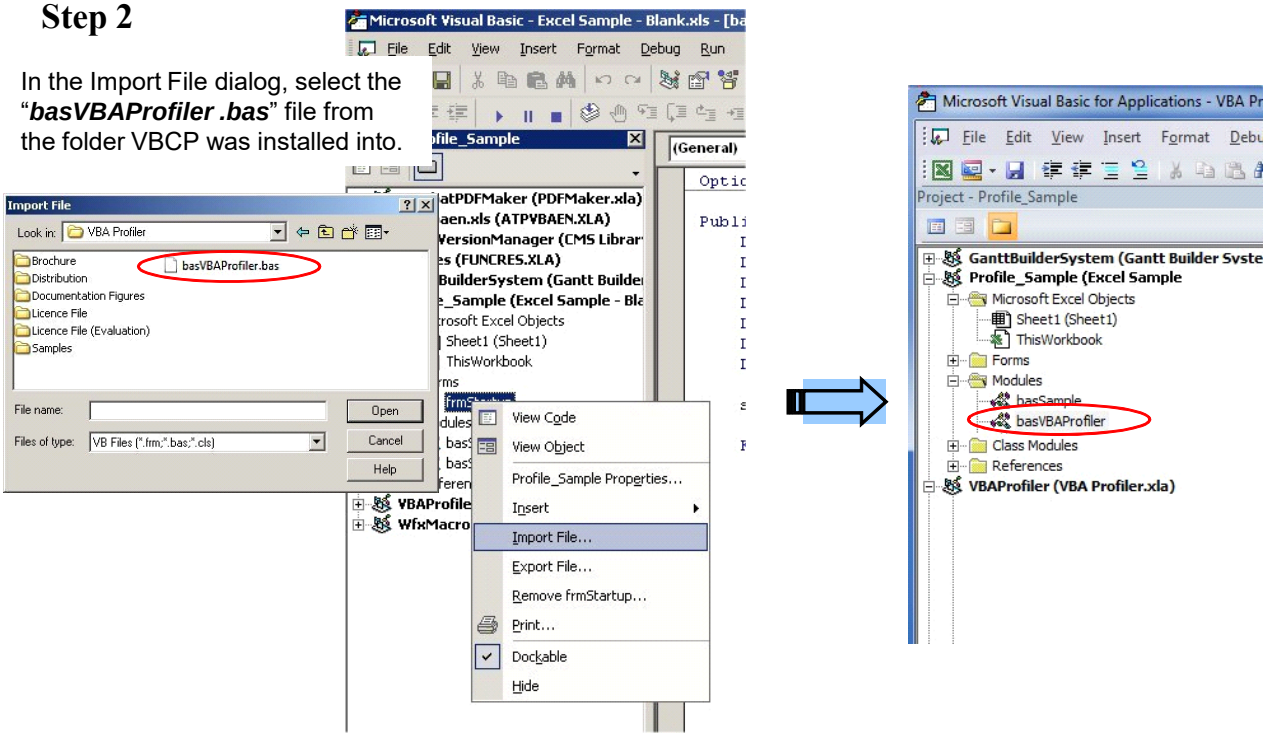


Figure 6 Instrumentation (Part 2)

then:

Step 3

After the module has been imported, view the code for this module (edit the #Const lines to match your system) and run the routine illustrated to the right to execute the assistant form, illustrated below:

Step 4

To add the instrumentation to your code, select the "***Profile Code"*** button. After instrumentation the module "***basProfilerConst***" will be created.

When you no longer require the code to be profiled, then repeat step 3 and select the ***"Clean Code"*** button.

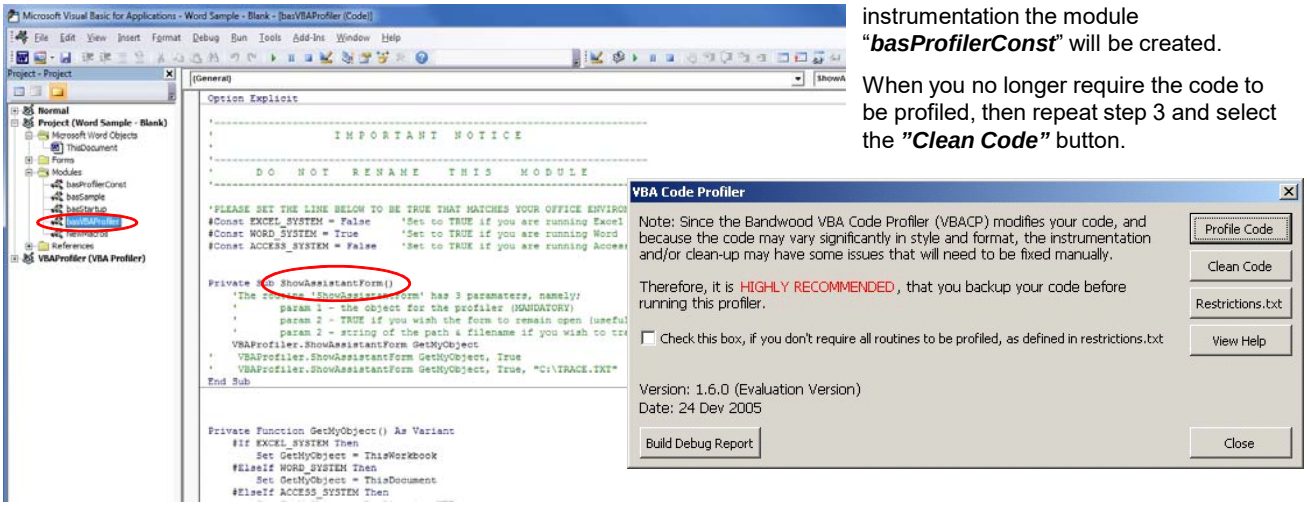


Figure 7 Instrumentation (Part 3)

Note: to save the instrumentation, you will need to manually save each modified project.

3.1.2.2 The Code Instrumentation Window

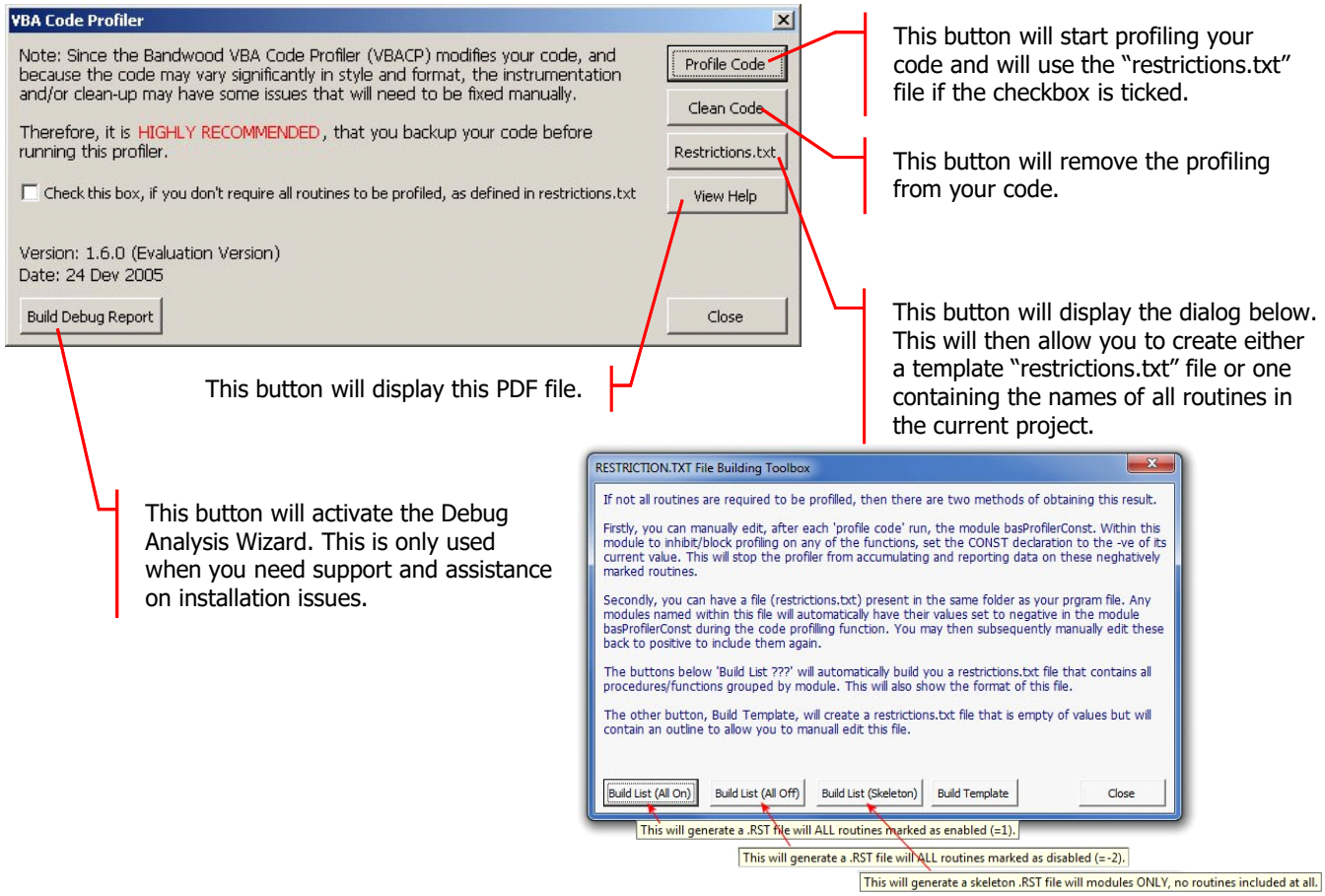


Figure 8 The Code Instrumentation Window

IMPORTANT – If the instrumentation (i.e. inserting or Profile your code) fails in any way then you will *NOT* be able to use "CLEAN CODE". This will only function correctly if you have successfully profiled your code.

If you do get an error, then the best option is to simply close your file/project WITHOUT saving, or revert to your backup copy.

3.1.3 Modifying How the Profiler Operates on Your Code

When profiling your code, you have two methods of restricting or modifying the behaviour of how the profiler performs.

3.1.3.1 Editing "basProfilerConst" Module

The first method is done entirely after you have profiled your code. This control is done via editing the **basProfilerCont** module created by the profiler.

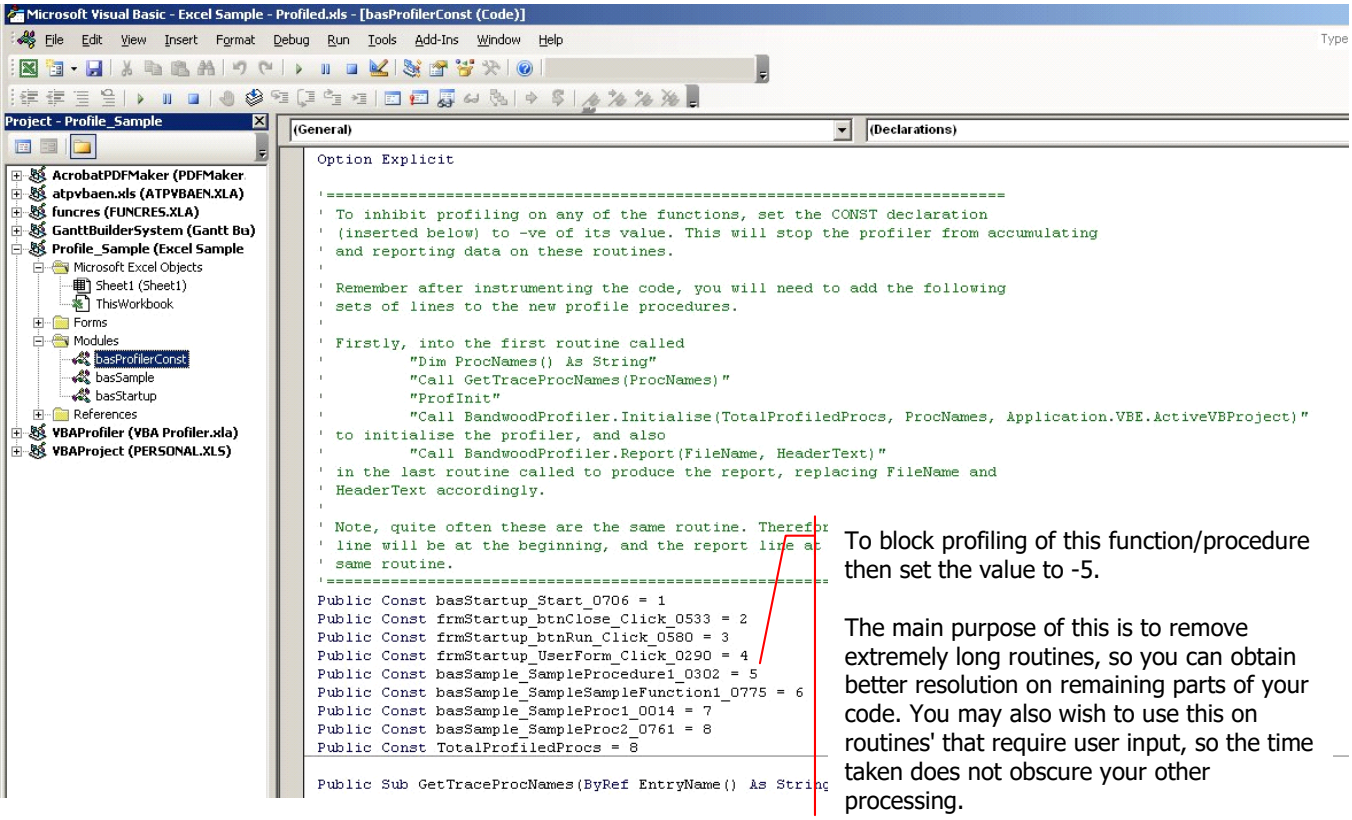


Figure 9 Manually Editing "basProfilerConst" Module

3.1.3.2 The "ProjectName.rst" File Format (Restrictions File)

The second method is done before you have profiled your code. This control is done via editing the **projectname.rst** file created by the profiler (note, this is a text file and can be edited with notepad or similar editor). See below for full details:

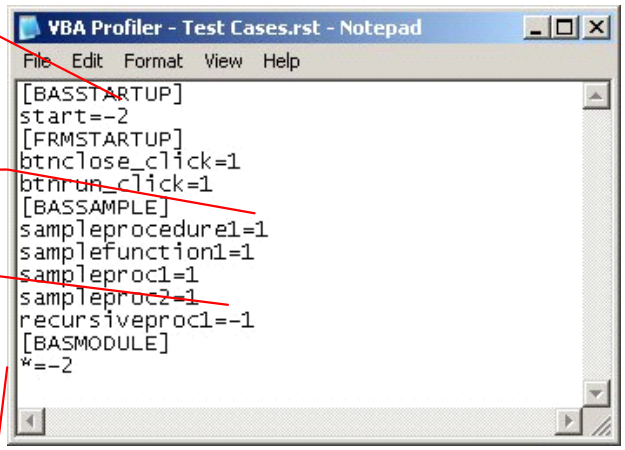
Below is a sample of the Restrictions file, name will be *project name* . RST

With a value of -2 against the routine, will cause the profiler to completely ignore this routine and leave it untouched within your code.

With a value of 1 against the routine, will cause the profiler to process normally this routine.

With a value of -1 against the routine, will cause the profiler to process normally this routine, but will mark it within the module basProfilerConst with a negative value to block profiling during the run. You may reset this back to positive ant any time.

You may also use an '*' asterisk to indicate all routines within a specific module, as illustrated here.



The overall structure of this file is each module will be UPPERCASE enclosed in square brackets. This defines the start of a section (ir module name), then the following procedure/functions names (in lowercase) as above.

Figure 10 The Restriction File Format

3.1.4 Library Reference Locations

3.1.4.1 Microsoft Word

The Microsoft® Word® library reference should be to the file "***VBA Profiler.dot***".

3.1.4.2 Microsoft Excel

The Microsoft® Excel® library reference should be to the file "***VBA Profiler.xla***".

3.1.4.3 Microsoft Access

3.1.4.3.1 Microsoft Access 2000-2003

The Microsoft® Access® library reference should be to the file "***VBA Profiler.mde***".

There is a known limitation with Access 2002/2003 version databases. See **Error! Reference source not found.** below for details.

3.1.4.3.2 Microsoft Access 2007-2016

The Microsoft® Access® library reference should be to the file "***VBA Profiler.accde***".

There is a known limitation with Access 2007-2016 version databases. See **Error! Reference source not found.** below for details.

3.2 PROFILING YOUR CODE

3.2.1 Setting-Up What to Profile

After you have instrumented your code, you need to add a number of lines manually to the routine/function that starts your program, if you want to profile the entire program. These lines are detailed in the comments section of the "***basProfilerConst***" module. Below are two examples:

Please refer to the detailed comments in the comments section of the "basProfilerConst" module for full information on setting up your profiler.

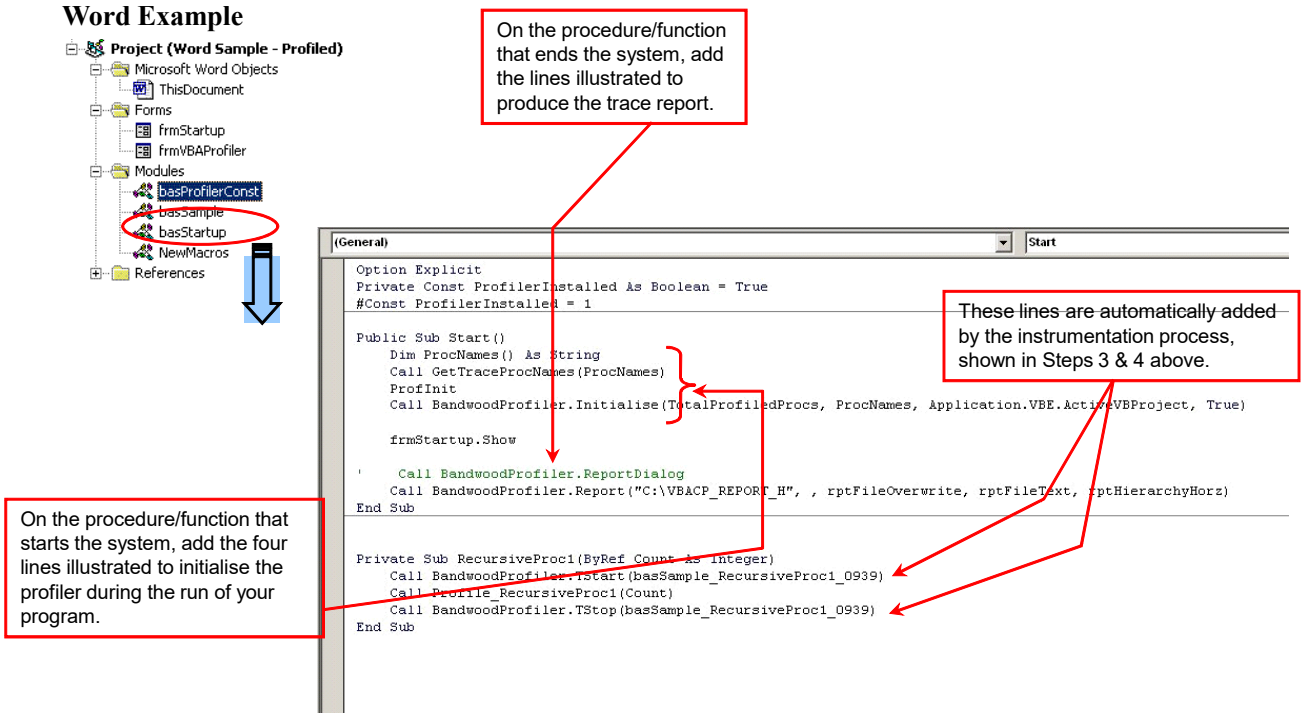


Figure 11 Adding the Initialisation & Report commands

If you only require profiling a specific part of your application, then insert the two lines indicated in the routine/function you wish to start profiling from.

You also have control over whether or not to construct and report on the procedural calling tree. This is controlled via an optional parameter (4th parameter) on the Initialise routine when the profiler is initialised, see above figure for example. The default, if this parameter is not provided, will be False (ie no tree report will be produced).

3.2.2 Required Visual Basic Editor Configuration

To ensure your code is profiled correctly, you need to ensure you have the VBE error traps set correctly. You should ensure "***Break on Unhandled Errors***" is selected is selected. If you have other errors, then the profiler may stop and report an error.

See the figure below:

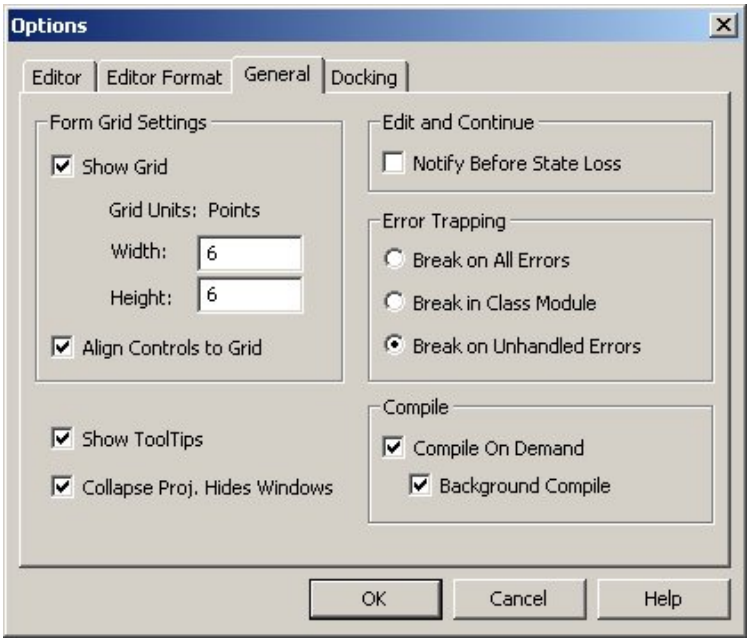


Figure 12 Visual Basic Environment Error Settings

If you do not have the setting correctly, as above, then the following is a typical error you will see;

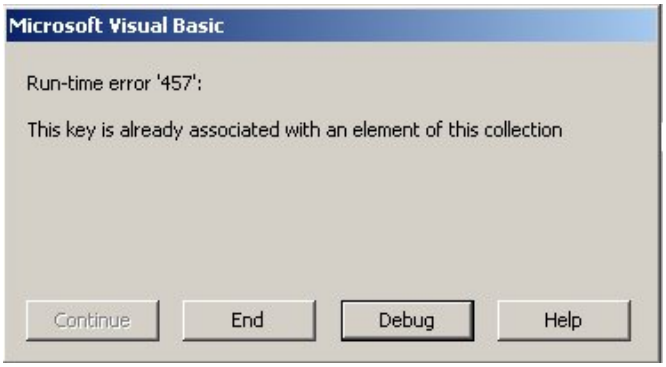


Figure 13 Visual Basic Errors

3.3 IMPORTANT NOTE

It is strongly recommended NOT to use the VBA Code Debugger during a profiling operation. This adversely affects the results generated in addition to causing issues with the creation/destruction of the profiling VBA object.

3.4 PROFILE REPORTS

There are two basic formats of output of profiled data. The first format is the standard tabulated results showing the total number of calls and time taken for each. The second format is a hierarchical tree showing the calling sequence and times for each node. For the hierarchical tree format, you may also choose a horizontally based layout or a vertically based layout.

Within each report type, you have an option of outputting in three different formats, standard textual format; comma separated variable CSV format, and HTML format. Note, a sample of each format is provided in the ***'Samples'*** folder of the installation.

3.4.1 Generating Reports

There are two basic ways of generating reports. One is via the Reports Dialog box. This is activated by inserting the following code fragment into your routine;

Call BandwoodProfiler.ReportDialog(Optional AutoLoad),
where if AutoLoad :=True will automatically load your saved settings (default:= False)

or the following to generate the report directly without user intervention;

Call BandwoodProfiler.Report(FileName, HeaderText, FileAction, FileType, HierarchyType, SummaryType)

You also have the ability to configure the CSV field separator (default is "," comma) but can be set to anything you like via the *BandwoodProfiler.CVSCharacter* property.

3.4.1.1 Dialog Prompt for Report Types

This dialog allows you to create multiple report styles and formats using the same execution data. See below for a sample of this dialog;

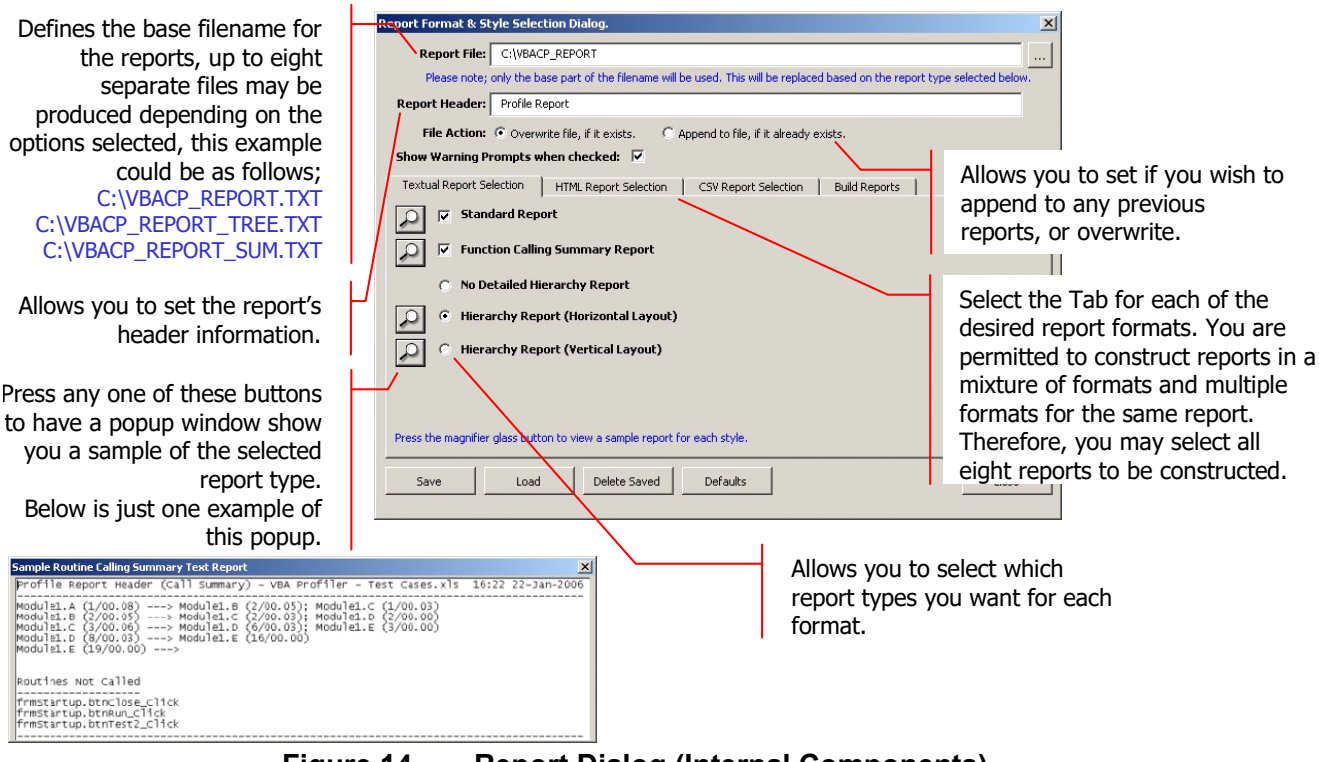


Figure 14 Report Dialog (Internal Components)

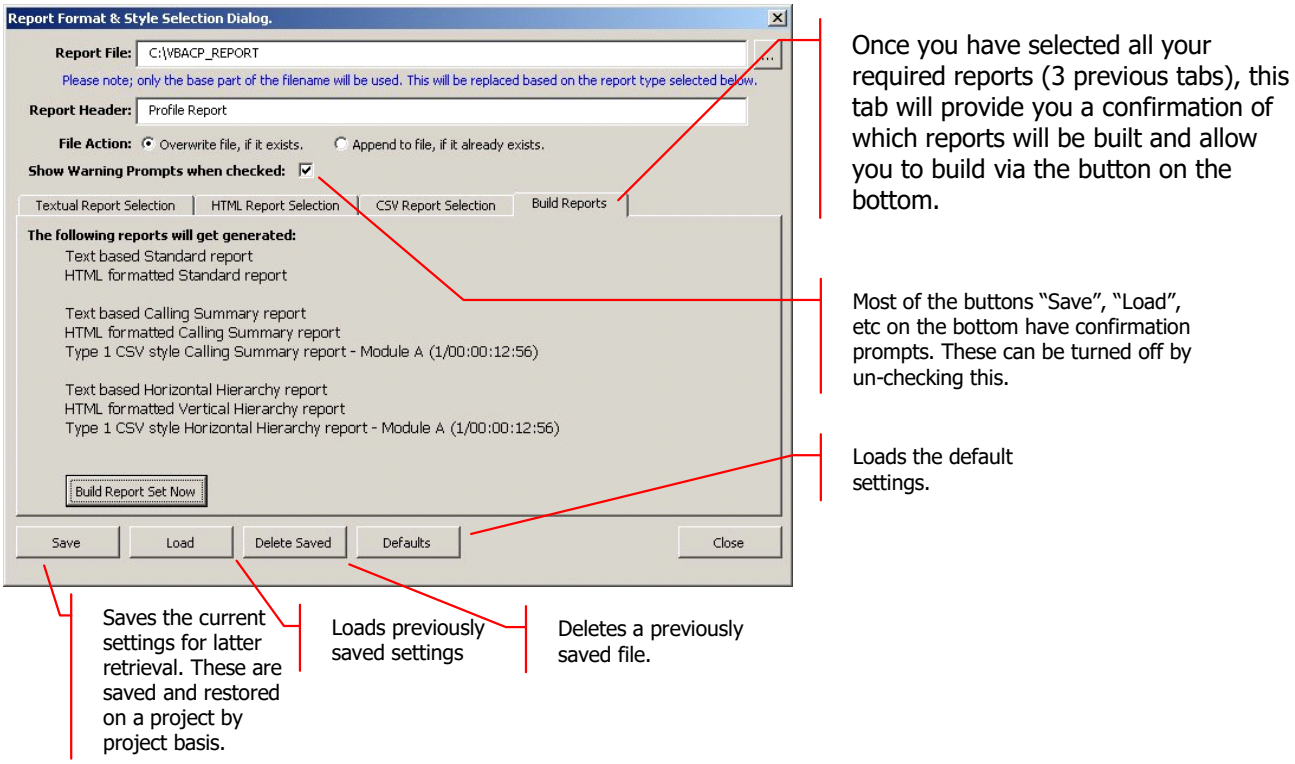


Figure 15 Report Dialog (Internal Components) – Cont.

3.4.1.2 Direct Programmatic Access

The function "***BandwoodProfiler.Report***" takes five optional parameters;

- ***FileName*** - the name of the report file (default:- "C:\VBA_Profiler.txt");
- ***HeaderText*** - the label on top of the report (default:- "Profile Report");
- ***FileAction*** - defines file overwrite action (default:- *rptFileAppend*). Below are the list of available constants for this variable;
    - *rptFileAppend* - append this report to the file, if one exists
    - *rptFilePrompt* - ask what action is required if the file already exists
    - *rptFileOverwrite* - replace the existing file
- ***FileType*** - defines file type of output file (default:- rptFileText), Below are the list of available constants for this variable;
    - *rptFileText* - creates a standard text file
    - *rptFileHTML* - creates HTML file
    - *rptFileCSV* - creates a type 1 comma separated variables (CSV) file
    - *rptFileCSV2* - creates a type 2 comma separated variables (CSV) file
- ***HierarchyType*** - defines file type of Hierarchy report (default:- rptHierarchyVert). Below are the list of available constants for this variable;
    - *rptHierarchyNone* - does not create any hierarchy report
    - *rptHierarchyVert* - creates a vertical based-out hierarchy report
    - *rptHierarchyHorz* - creates horizontal based hierarchy report

- ***SummaryType*** - defines file type of calling summary output file (default:-rptSummaryNone), Below are the list of available constants for this variable;
    - *rptSummaryNone* - does not create any calling summary file
    - *rptSummaryText* - creates a standard text calling summary file
    - *rptSummaryHTML* - creates a HTML calling summary file
    - *rptSummaryCSV* - creates a type 1 CSV calling summary file
    - *rptSummaryCSV2* - creates a type 2 CSV calling summary file

3.4.1.3 Difference between Type 1 CSV and Type 2 CSV

3.4.1.3.1 Type 1 CSV

Type 1 CSV file has only one separator character between each field routine field set, see below for an example;

Module1.A (1/00.08),

3.4.1.3.2 Type 2 CSV

Type 2 CSV file has only multiple separator characters for each field routine field set, see below for an example;

Module1.A , 1 , 00.08 ,

3.4.2 Interpreting the Standard Profile Report

The output below is in the form of a text based report to allow importing into Excel and other tools for graphing purposes in the CSV format. The profile report produced by the system is detailed below:

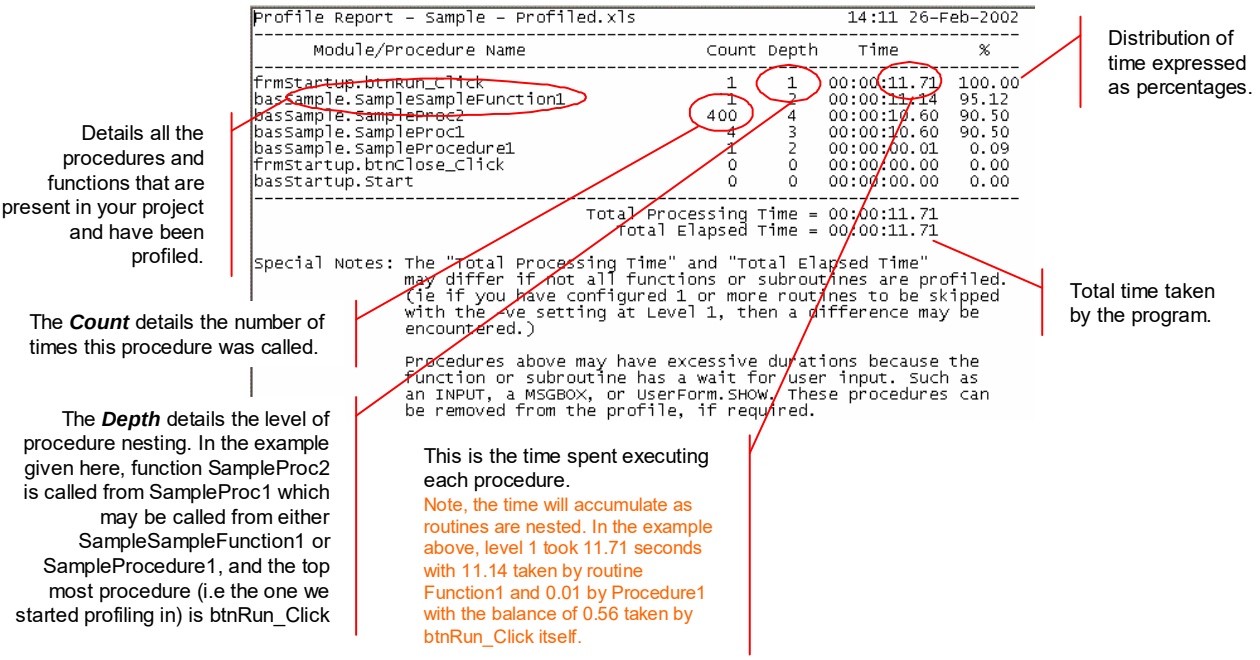


Figure 16 Standard Report Interpretation

Special Note: if you're routine prompts the user for input (i.e. a MsgBox) within a profiled routine, then the time recorded spent in that routine will include the time taken by the user to complete the prompt and continue.

3.4.3 Interpreting the Hierarchical Profile Report

The output below is in the form of a hierarchical based report in vertically layout HTML format. The profile report produced by the system is detailed below:

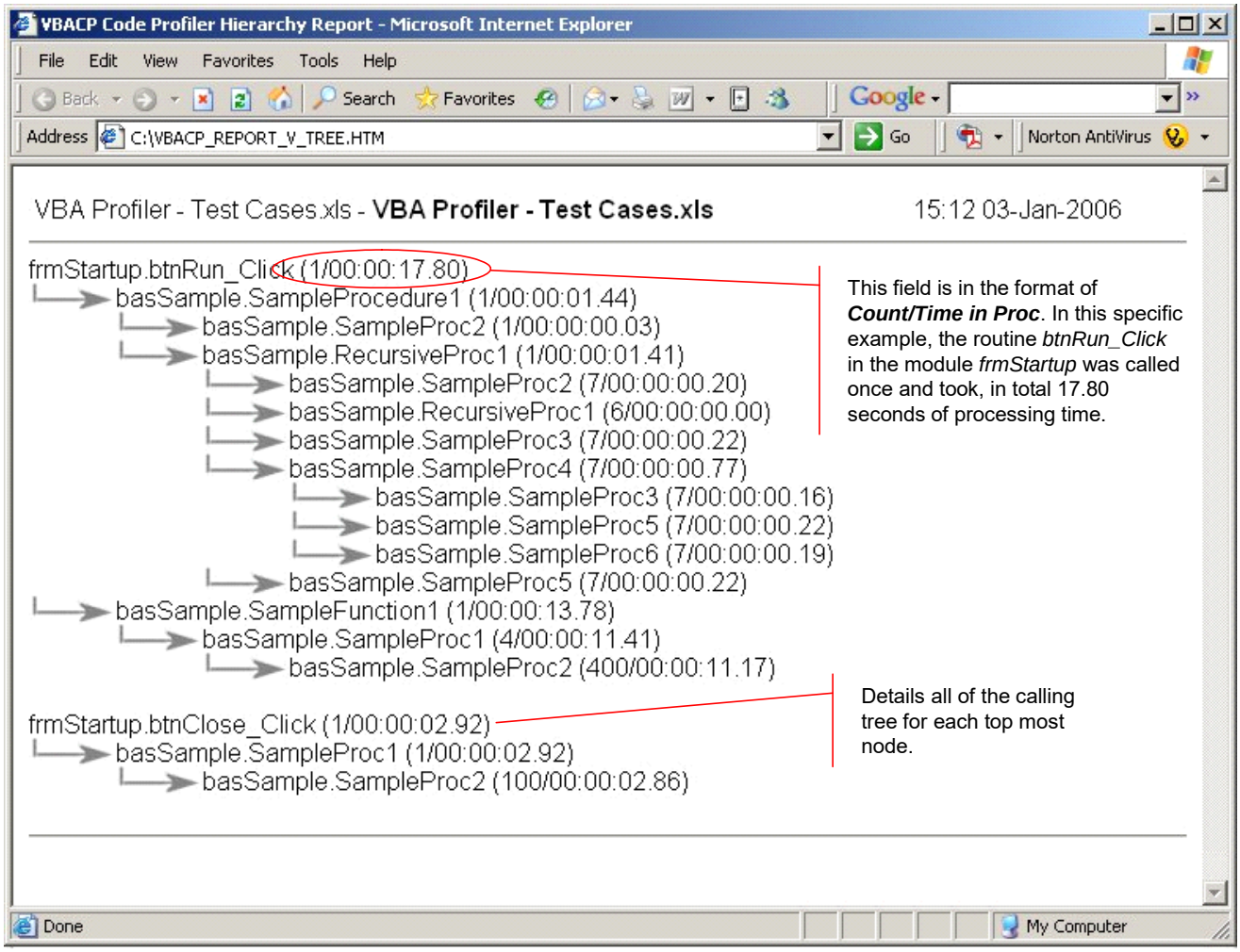


Figure 17 Hierarchical Report Interpretation

3.4.4 Interpreting the Calling Summary Profile Report

The output below is in the form of a calling summary report in HTML format. The profile report produced by the system is detailed below:

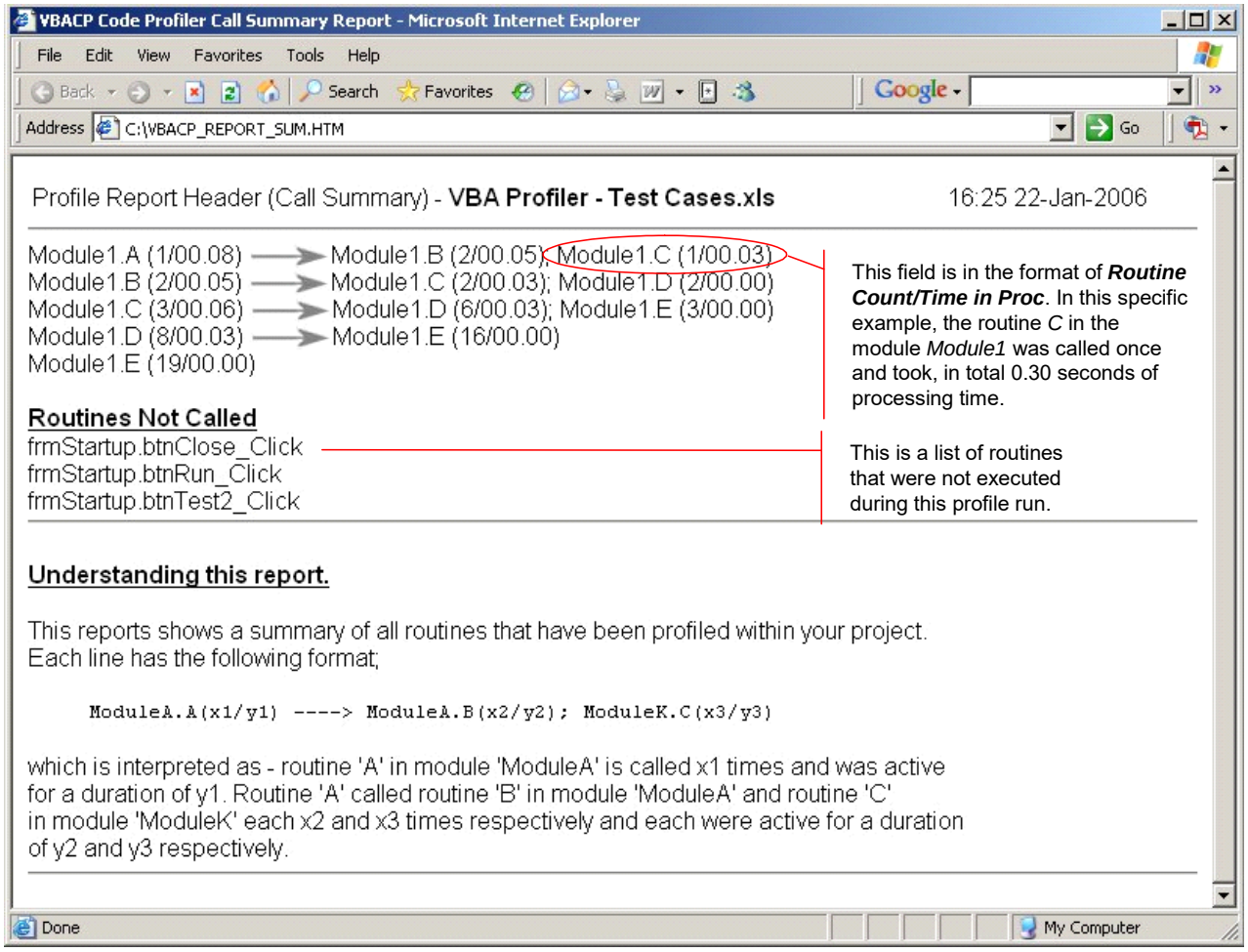


Figure 18 Hierarchical Report Interpretation

3.4.5 Report Capacity Warning

On large projects or projects with very large call depths, the report engine may generate an overflow error within itself (ie *Procedure "HierarchyReport" Error 6 -> Overflow*). This is normally associated with the HTML formatted reports. Try either Text or CSV formatted report. If the error still occurs, then reduce the size of your profiled code by using the techniques described in Section 3.1.3 - Modifying How the Profiler Operates on Your Code.

3.4.6 Report Accuracy Warning

To ensure the accuracy of the profiled times, the PC running the profiler MUST have been rebooted within the last 49 days. If your PC has been running longer than 49 days, then you may get incorrect timing information. This is necessary to ensure millisecond level timing.

3.5 CODE CLEANING

3.5.1 Manual Code Cleaning

To remove the instrumentation from your code, you will need to follow Steps 3 & 4 from above. After the cleaning of your code, you will need to save the project.

3.6 REMOVING THE PROFILER COMPLETELY

To remove the VBA code profiler completely from your code, you must first clean your code and then remove the module "***basVBAProfiler***" from your project and finally remove the reference added to the library, then save.

4 PROGRAM LIMITATIONS

4.1 EVALUATION LIMITATIONS

There is an evaluation limit on the number of procedures/functions and the number of modules that can be profiled. The evaluation limitations are detailed below:

Evaluation Version	
Limit Per Module	**Limit per Project**
50	250

Special note, if you intend to use the extended limitations, and then please ensure your hardware has sufficient memory and CPU power to perform the operations. Memory errors are not catered for and we stress backup your projects before instrumenting your code.

4.2 KNOWN PROFILER LIMITATIONS

There is an implementation limit on the number of procedures/functions and the number of modules that can be profiled. The standard/default limitations are detailed below:

Evaluation Version		Fully Licensed Version	
Limit Per Module	**Limit per Project**	**Limit Per Module**	**Limit per Project**
50	250	500	5000

However, for some very large projects, you can extend the standard limits by having a file named "***extend.yes***" present in the same folder as the library. With this file present the following limitations will apply:

Evaluation Version		Fully Licensed Version	
Limit Per Module	**Limit per Project**	**Limit Per Module**	**Limit per Project**
50	250	30,000	30,000

Special note, if you intend to use the extended limitations, and then please ensure your hardware has sufficient memory and CPU power to perform the operations. Memory errors are not catered for and we stress backup your projects before instrumenting your code.

4.3 CODE STYLE LIMITATIONS

Some code style limitations are also present, these are as follows:

- The profiler will not process routines expressed on a single line with continuation markers. For example:

  Private Sub Test(ByVal J As Integer): J=J+1: End Sub

- Interfaces are not recognised by the profiler (ie simply can't transform them). But since interfaces never contain any implementation, profiling them is not required. The best solution is to simply switch the whole interface module off manually, by adding a section to the restriction file (ProjectName.rst).such as;

  [InterfaceModuleName]

  ****=-2*** (that is negative 2)

  There are no plans at the moment to include interfaces as part of the profiler's capabilities in the foreseeable future.

- The Subroutine/Function name must appear on the same line as the declaration (ie the profiler will not cater for the following example);

  ```
  Private Function _
  TestFunc6(ByVal XYZ As Variant) As String
  ....
  ....
  TestFunc6 = "Return String"
  End Function
  ```

- Because the code is not compiled as part of the profiling process the VBA construct **#If...Then...#Else #EndIf** providing conditional compiling selected blocks of Visual Basic code is not supported. Unpredictable results may occur when these constructs are used. These will need to be manually edited after the profiler has completed. Note, code cleaning may also be adversely affected.

- Set/Let are not inferred by the profiler. This is a current limitation and there are no plans to include inferred assignments as part of the profiler's capabilities.

5 MAINTENANCE SUPPORT

If you have purchased annual maintenance support, then upgrades are available directly from our web site at http://www.bandwood.com/support.htm

You will require your password given when you take up the support to gain access to this site.

6 CONTACTS

If you have suggestions or have identified a defect, please e-mail all relevant details to support@bandwood.com. The item will be investigated and scheduled to be incorporated into the next release.

The staff at Bandwood Pty Limited hopes you enjoy using VBACP and find it beneficial to your business. We are constantly improving our software products and welcome feedback on all of our product range. Therefore, we would be pleased to receive suggestions to support@bandwood.com on this product.

7 APPENDIX A – CANNOT ADD LIBRARY REFERENCE

Sometimes, if your Excel or Word environment does not have the correct Office Security setting, then the following may occur;

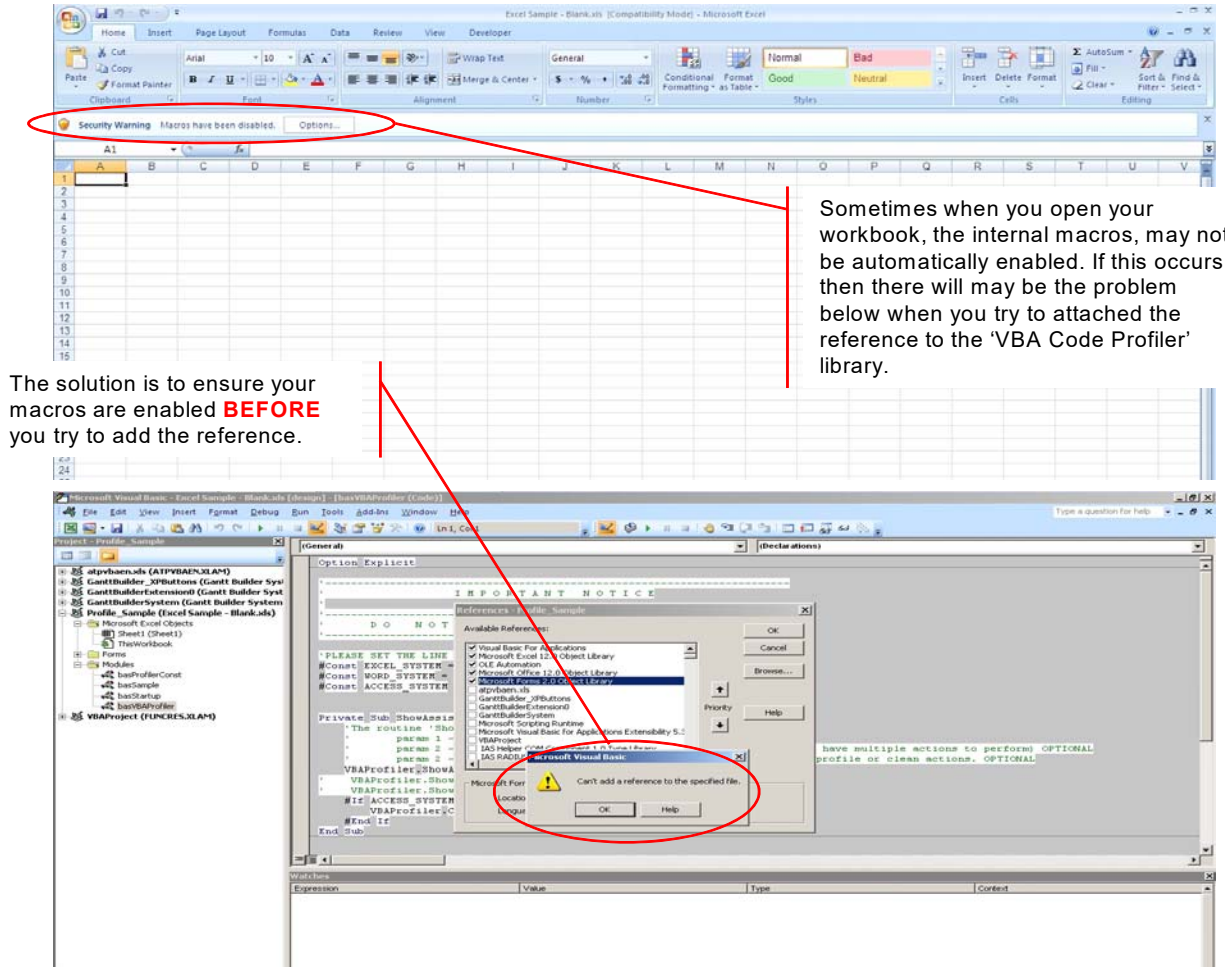


Figure 19 Problem Referencing Library Example