Debug Break Checker

by

Software Verify

SoftwareVerify

Troubleshoot your software

Debug Break Checker

__debugbreak usage detector for Windows Operating Systems

by Software Verify Limited

Welcome to the Debug Break Checker software tool. Debug Break Checker is software tool that scans source code looking for specific uses of __debugbreak(). Once you have found these specific uses you can decide if they are correct for the context in which they are found.

We hope you will find this document useful.

Debug Break Checker Help

Table of Contents

Foreword

This is just another title page
placed between table of contents
and topics

Part

I
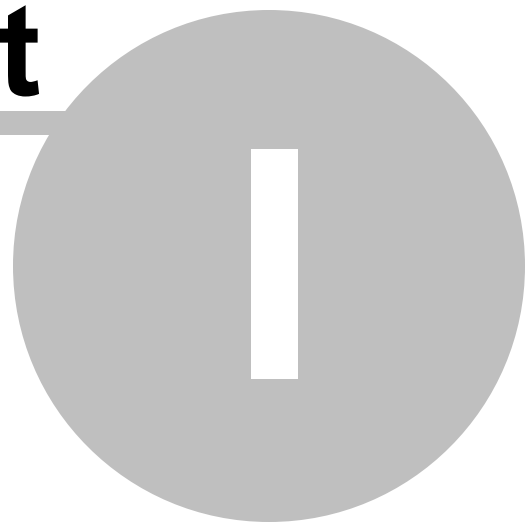
1 How to get Debug Break Checker

Debug Break Checker is free for commercial use. Debug Break Checker can be downloaded for Software Verify's website at https://www.softwareverify.com/product/debugbreak-checker/

This help manual is available in Compiled HTML Help (Windows Help files), PDF, and online.

Windows Help	https://www.softwareverify.com/documentation/chm/debugBreakChecker.chm
PDF	https://www.softwareverify.com/documentation/pdfs/debugBreakChecker.pdf
Online	https://www.softwareverify.com/documentation/html/debugBreakChecker/index.html

Whilst Debug Break Checker is free for commercial use, Debug Break Checker is copyrighted software and is not in the public domain.

You are free to use the software at your own risk.

You are not allowed to distribute the software in any form, or to sell the software, or to host the software on a website.

Contact Software Verify at:

Software Verify Limited
Suffolk Business Park
Eldo House
Kempson Way
Bury Saint Edmunds
IP32 7AR
United Kingdom

email	sales@softwareverify.com
web	https://www.softwareverify.com
blog	https://www.softwareverify.com/blog
twitter	http://twitter.com/softwareverify

Visit our blog to read our articles on debugging techniques and tools.
Follow us on twitter to keep track of the latest software tools and updates.

Part II
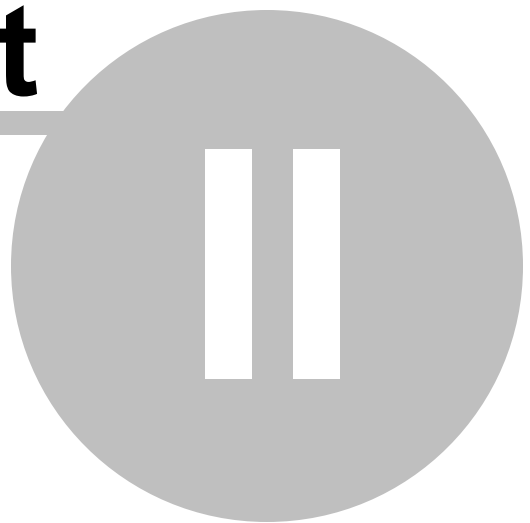
2   What does Debug Break Checker do?

Debug Break Checker scans your source code looking for calls to __debugbreak() and reports each call with a list of conditional logic or conditional compilation directives. This allows you to determine if a call to __debugbreak() has been left in the code by accident (and should thus be removed) or should be present in the code.

As well as checking for __debugbreak() calls, you can also check for any functions you choose to specify.

Part III
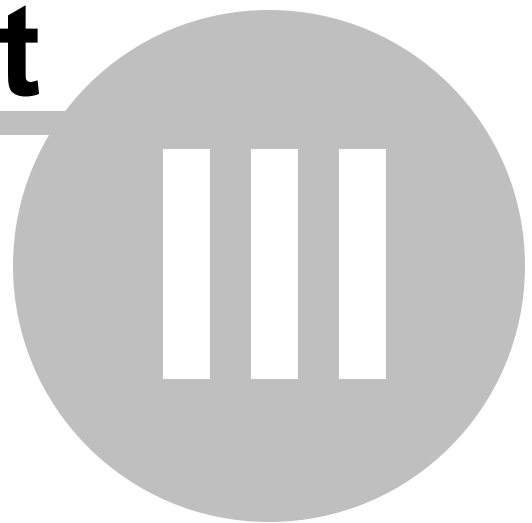
3   Default Conditionals

The default settings will cause Debug Break Checker to recognise the following conditional compilation statements:

    _DEBUG

What if I've got my own conditionals I need to check for?

If you've got your own conditionals you can specify these using the settings dialog or the command line.

Part IV

4 Default Functions

The default settings will cause Debug Break Checker to search for the following functions:

Win32

      __debugbreak

What if I've got my own functions I need to check for?

If you've got your own functions you can specify these using the settings dialog or the command line.

Part V

5   Menu

The main menu contains four menus, File, Edit, Software Updates and Help.
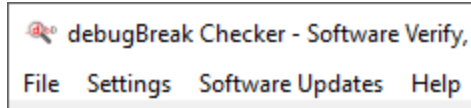
5.1   File

The File menu controls starting and stopping scans for calls to __debugbreak, clearing the display and exiting the program.
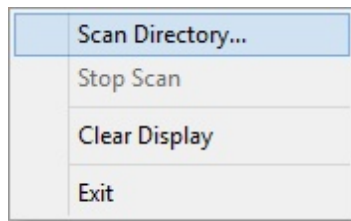
File menu ➡ **Scan Directory...** ➡ displays a directory chooser then starts scanning that directory for calls to the functions listed in the settings.
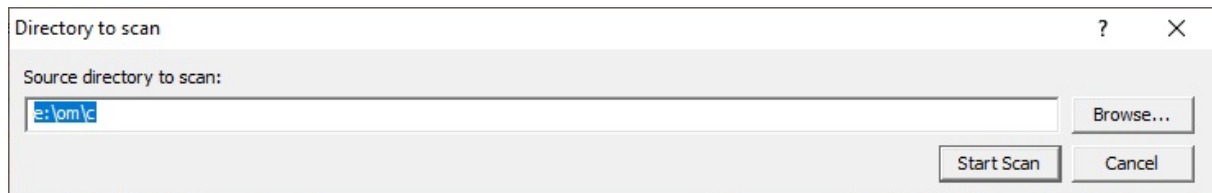
File menu ➡ **Stop scan** ➡ stops any scan that is in progress

File menu ➡ **Clear Display** ➡ clear all results from a previous scan and to remove any source code from the source code editing window.

File menu ➡ **Exit** ➡ closes Debug Break Checker

5.2   Edit

The Edit menu controls editing settings, loading settings and saving settings.

Edit menu ⇨ **Settings...** ⇨ displays the settings dialog.

Edit menu ⇨ **Load Settings...** ⇨ loads previously saved settings.

Edit menu ⇨ **Save Settings...** ⇨ saves settings.

5.3 Software Updates

The Software Updates menu controls how often software updates are downloaded.

If you've been notified of a new software release to Debug Break Checker or just want to see if there's a new version, this feature makes it easy to update.
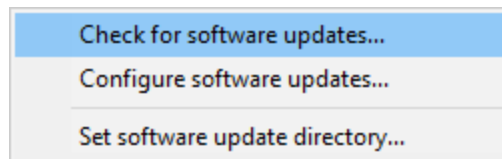
◫ **Software Updates** menu ⇨ **Check for software updates** ⇨ checks for updates and shows the software update dialog if any exist
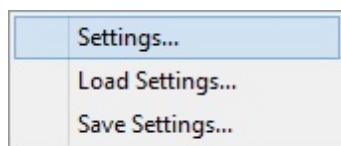
An internet connection is needed to be able to make contact with our servers.

🖳 Before updating the software, close the help manual, and end any active session by closing target programs.

If no updates are available, you'll just see this message:
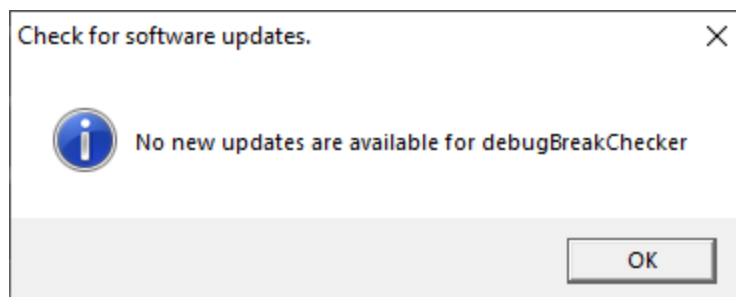
Software Update dialog

If a software update is available for Debug Break Checker you'll see the software update dialog.

Software update download confirmation ✕

A software update is available for debugBreakChecker.

Do you wish to install debugBreakChecker V1.27?

| Yes, Download and Install | Don't download, ask me later | Skip this version, don't tell me about it again |

Software update

Software update options...

- **Download and install** ⇨ downloads the update, showing progress

Downloading debugBreakChecker 1.27 ✕

Downloading debugBreakChecker 1.27

[progress bar] Stop

5408 KB of 8728 KB (62.0%)
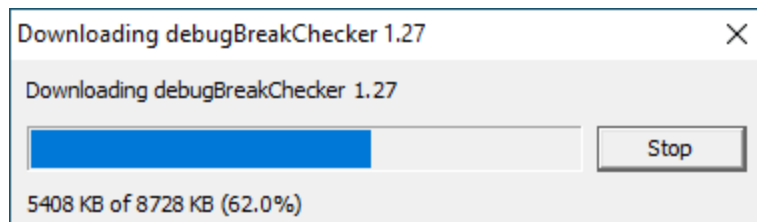
Once the update has downloaded, Debug Break Checker will close, run the installer, and restart.

You can stop the download at any time, if necessary.

- **Don't download...** ⇨ Doesn't download, but you'll be prompted for it again next time you start Debug Break Checker

- **Skip this version...** ⇨ Doesn't download the update and doesn't bother you again until there's an even newer update

- **Software update options...** ⇨ edit the software update schedule

Problems downloading or installing?

If for whatever reason, automatic download and installation fails to complete:

- Download the latest installer manually from the software verify website.

Make some checks for possible scenarios where files may be locked by Debug Break Checker as follows:

- Ensure Debug Break Checker and its help manual is also closed

- Ensure any error dialogs from the previous installation are closed

You should now be ready to run the new version.

Software update schedule

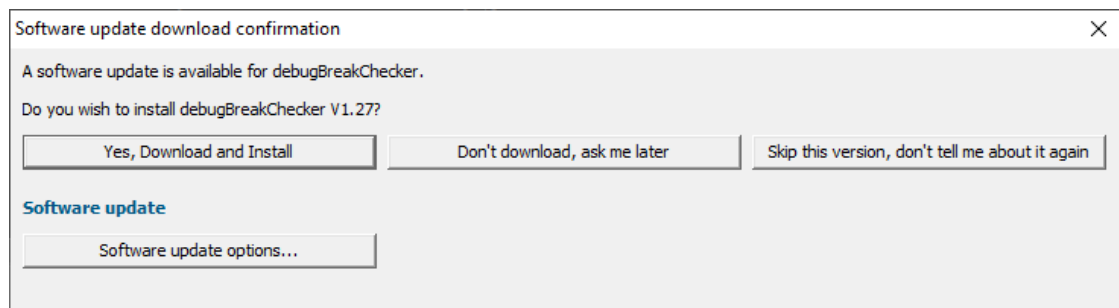Debug Break Checker can automatically check to see if a new version of Debug Break Checker is available for downloading.

    ▤ **Software Updates** menu ⇨ **Configure software updates** ⇨ shows the software update schedule dialog

The update options are:

- never check for updates
- check daily (the default)
- check weekly
- check monthly

The most recent check for updates is shown at the bottom.
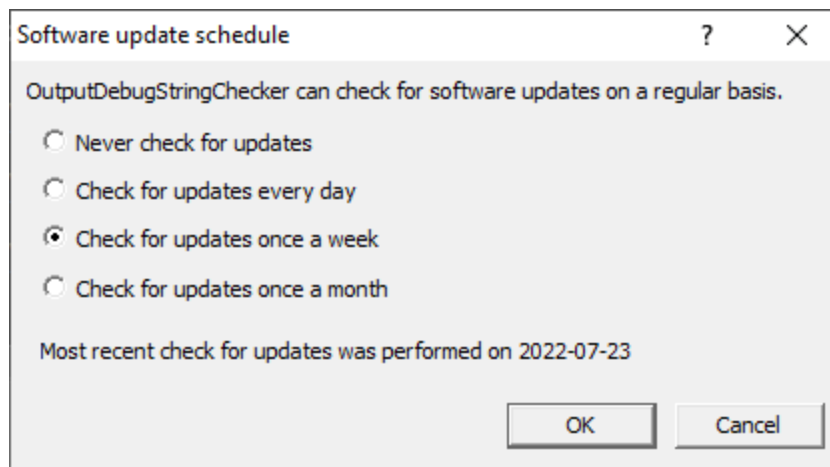
Software update directory

It's important to be able to specify where software updates are downloaded to because of potential security risks that may arise from allowing the `TMP` directory to be executable. For example, to counteract security threats it's possible that account ownership permissions or antivirus software blocks program execution directly from the `TMP` directory.

The `TMP` directory is the default location but if for whatever reason you're not comfortable with that, you can specify your preferred download directory. This allows you to set permissions for `TMP` to deny execute privileges if you wish.

    ▤ **Software Updates** menu ⇨ **Set software update directory** ⇨ shows the Software update download directory dialog
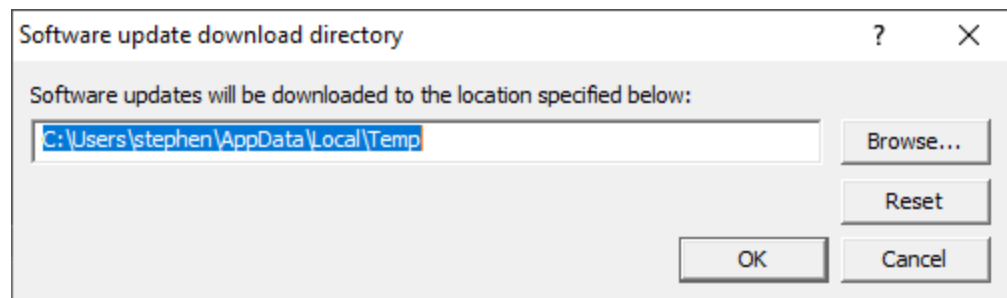
An invalid directory will show the path in red and will not be accepted until a valid folder is entered.

Example reasons for invalid directories include:

- the directory doesn't exist
- the directory doesn't have write privilege (update can't be downloaded)
- the directory doesn't have execute privilege (downloaded update can't be run)

When modifying the download directory, you should ensure the directory will continue to be valid. Updates may no longer occur if the download location is later invalidated.

- **Reset** ⇨ reverts the download location to the user's `TMP` directory

  The default location is `c:\users\[username]\AppData\Local\Temp`

5.4 Help

The Help menu controls displaying this help document and displaying information about Debug Break Checker.
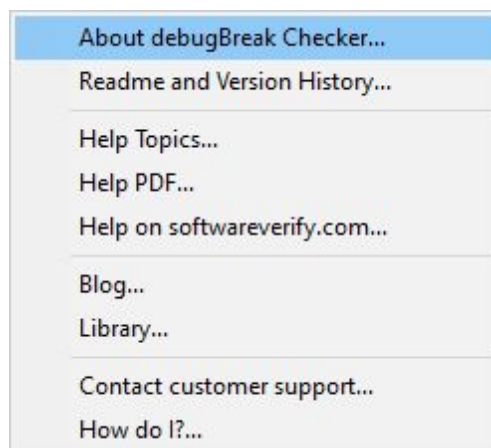
Help menu ⇨ **About Debug Break Checker...** ⇨ displays information about Debug Break Checker.

Help menu ⇨ **Readme and Version History...** ⇨ displays the readme and version history.

Help menu ➡️ **Help Topics...** ➡️ displays this help file.

Help menu ➡️ **Help PDF...** ➡️ displays this help file in PDF format.

Help menu ➡️ **Help on softwareverify.com...** ➡️ display the Software Verify documentation web page where you can view online documentation or download compiled HTML Help and PDF help documents.

Help menu ➡️ **Blog...** ➡️ display the Software Verify blog.

Help menu ➡️ **Library...** ➡️ display the Software Verify library - our best blog articles grouped by related topics.

Help menu ➡️ **Contact customer support...** ➡️ displays the options for contacting customer support.

Contact Software Verify Customer Support   ✕

We provide customer support via email.

Email allows us to exchange detailed bug descriptions, detailed instructions, screenshots, log files, crash dumps and other metadata that can't easily be communicated via telephone or chat.

Should your support request require escalation to phone, Zoom or remote computer access we will do that when required.

There are two methods to start a support request:

Email:

support@softwareverify.com

Website customer support form:

https://www.softwareverify.com/support.php

Close

Click a link to contact customer support.

Help menu ➡️ **How do I?...** ➡️ displays the options for asking us how to do a particular task.

How do I? ✕

How do I fly to the moon?

How do I make a purple cow?

How do I do XYZ with debugBreakChecker?

We can't help with the first two questions, but I think we can help with the last question.

We'll give you step by step instructions and/or a video showing you how to do XYZ.

Just send us an email describing what you're trying to do and we'll get right back to you with a solution.

Start a Support Request

Email:

support@softwareverify.com

Website customer support form:

https://www.softwareverify.com/support/

Close

Part VI

6 The user interface

The Debug Break Checker user interface is split into three sections:

- Menu
- Error report grid
- Source code viewer

Error Report Grid

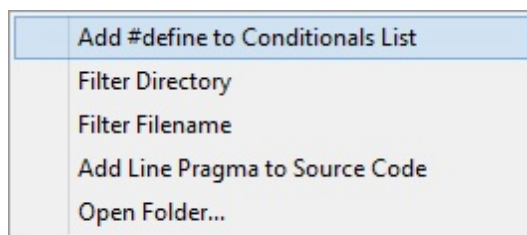The error report grid displays one warning per line. Each line contains the test function name, any conditional compilation values in force, the name of the function containing the test function, the line number and the filename of the error.

You can sort the data by clicking on any column header. Click again to reverse the sort direction.

Click on any line to display the source code in the lower window. Double click to edit the source code in Visual Studio.

Test	Conditions	Function	Line Nu...	Filename
__debugbreak	if (MessageBox(NULL, message, _T("Attach Debugger...	void initPlumberServer_d...	3442	E:\om\c\memory32\stub\PLUMINIT.CPP
__debugbreak	if (stubSettingData::doBreakContinue(_T("The process...	static NAKEDDEF VOID h...	6501	E:\om\c\memory32\stub\future\HandleKernelBaseHook.cpp
__debugbreak	if (offset == i && enableBreakpoints)	static void *findFunction(...	858	E:\om\c\memory32\stub\libCFunctionHook.cpp
__debugbreak	if (iter != purgedTraces.end())	void Session::checkStack...	56232	E:\om\c\memory32\tabserv\session.CPP
__debugbreak	if (MessageBox(NULL, _T("PLUMINIT.CPP: Attach Deb...	void initPlumberServer_d...	2647	E:\om\c\performanceValidator\stub\PLUMINIT.CPP
__debugbreak	if (Id3 != NULL)	void CEditTextWnd::debu...	8610	E:\om\c\research\tooltips\edittool\D_EDITT2.CPP
__debugbreak	if (Id1 == NULL)	void CEditTextWnd::debu...	8615	E:\om\c\research\tooltips\edittool\D_EDITT2.CPP
__debugbreak	if (MessageBox(NULL, _T("Debug this? svlApplication...	extern "C" __declspec(dll...	90	E:\om\c\svlApplicationToMonitor\svlApplicationToMonitor\svlA
__debugbreak	if (MessageBox(NULL, _T("Debug this? svlApplication...	extern "C" __declspec(dll...	127	E:\om\c\svlApplicationToMonitor\svlApplicationToMonitor\svlA
__debugbreak		void stubSettingData::do...	161	E:\om\c\svlBug\stubSettingData.cpp
__debugbreak	if (MessageBox(NULL, _T("Drop into the debugger?\r\...	void commandLine::inter...	715	E:\om\c\svlCommonMFC\commandLine.cpp
__debugbreak	if ((ePtr[0] == 0xFF) && (ePtr[1] == 0x51) && debug...	int FunctionHooker::doA...	5848	E:\om\c\svlCommonStub\FunctionHooker.cpp
__debugbreak	if ((ePtr[0] == 0xFF) && (ePtr[1] == 0x51) && debug...	int FunctionHooker::doA...	5989	E:\om\c\svlCommonStub\FunctionHooker.cpp
__debugbreak	if ((iPtr[0] == 0xFF) && (iPtr[1] == 0x51) && debugE...	int FunctionHooker::doA...	6095	E:\om\c\svlCommonStub\FunctionHooker.cpp
__debugbreak	if (MessageBox(NULL, _T("Address is within range of i...	void vmProtect::testDebu...	477	E:\om\c\svlCommonStub\vmProtect.cpp

A context menu is available on the error report grid.

Add #define to Conditionals List
Filter Directory
Filter Filename
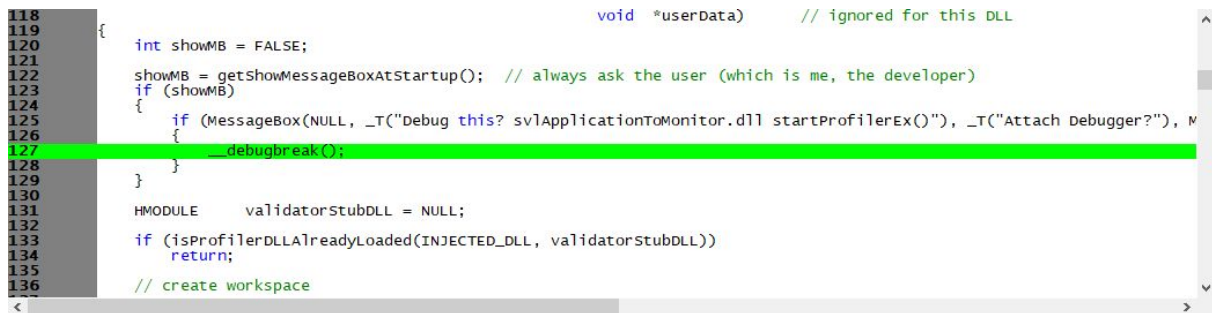Add Line Pragma to Source Code
Open Folder...

The context menu has five options:

- An option to add the conditional compilation #define value to the list of known conditional compilation values. Any values set can be edited on the Conditionals settings.

- Two options for filtering by directory and by filename. Any values set by these filters can be edited on the File Filters settings.

- An option to insert a line pragma at the end of the selected line. Any values set can be edited on the Line Pragma settings.

- An option to open an explorer window at the directory for the selected line.

Source Code Viewer

The source code viewer displays the source code with the line of interested highlighted in green.
Click on any line in the error report grid to view the source code in the source code viewer.
If you would rather edit the source code using Visual Studio, double click the line in the grid.
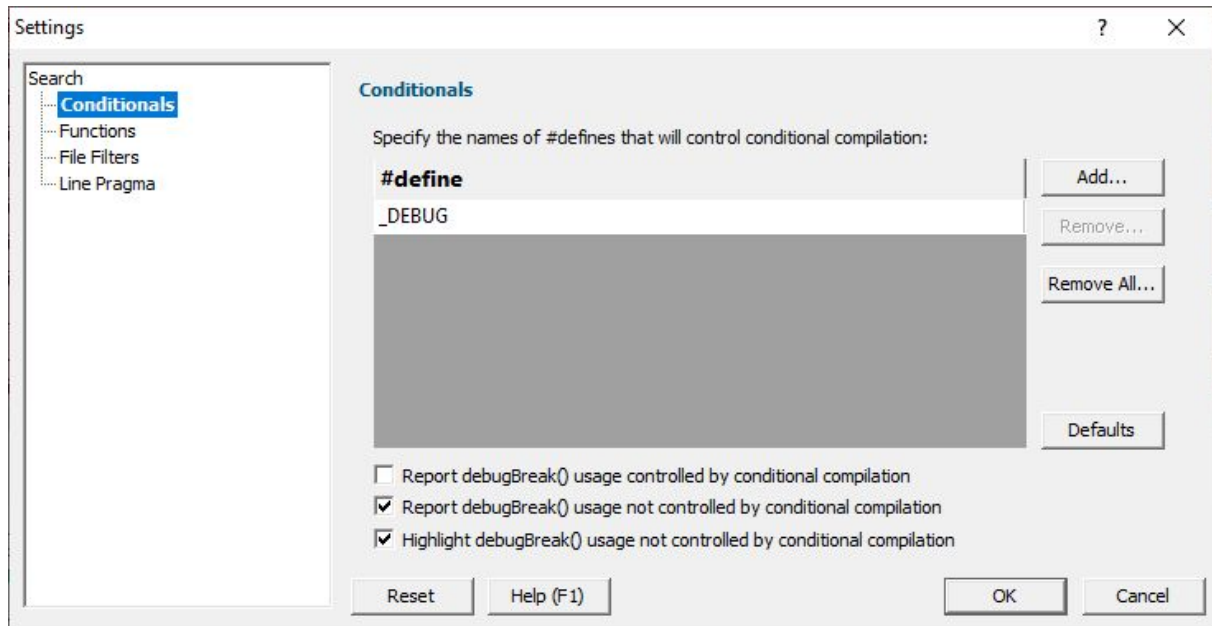
Part VII

7 Modifying the settings

The settings dialog allows you to control all Debug Break Checker settings.

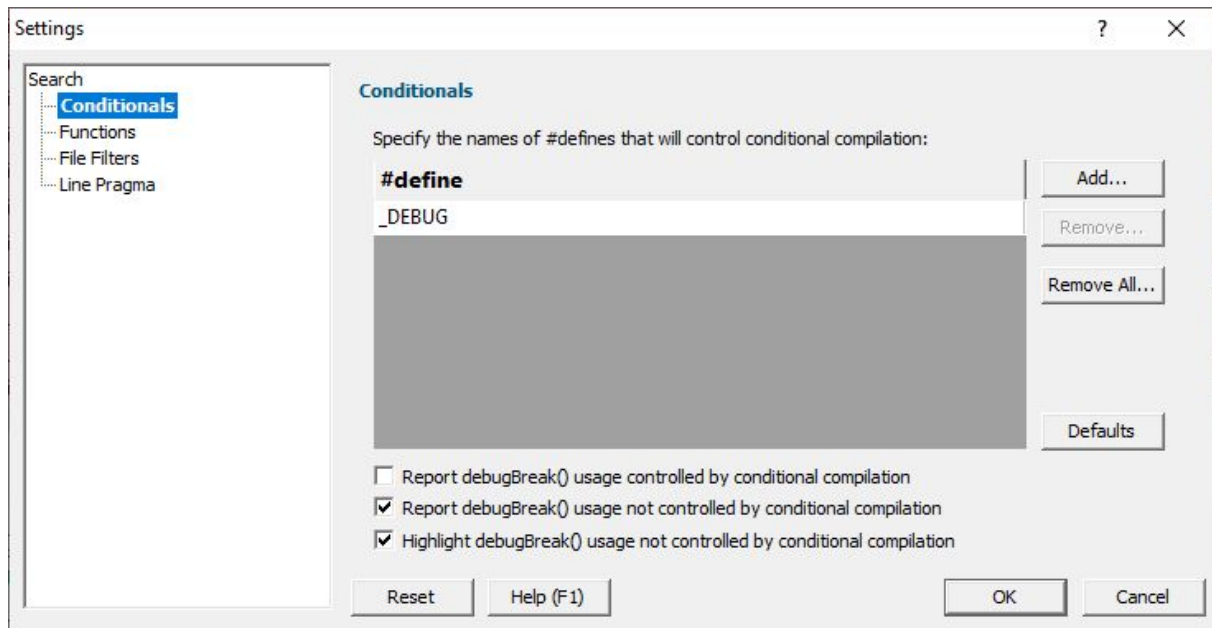The settings are grouped into logical choices making it easier to manage the settings.

Reset
Click **Reset** to reset all settings on all pages of the settings dialog.

7.1 Conditionals

The Conditionals page of the settings dialog allows you to control how OutputDebugStringChecker responds to conditional compilation of C and C++ source code.
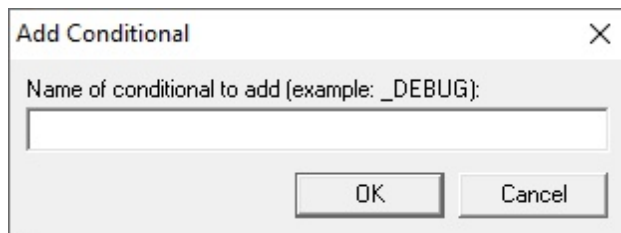
Conditionals

You can add and remove conditional compilation defines that OutputDebugStringChecker will recognise.

Add
Click **Add** to add a new conditional.

The Add Conditional dialog box will be displayed. Type the name of the new conditional and click OK.

Remove
Click **Remove** to remove the selected conditionals.

Remove All
Click **Remove All** to remove all conditionals.

Defaults
Click **Defaults** to restore the default conditionals.

Conditional Compilation

You can change what OutputDebugStringChecker reports by using the two **Report __debugbreak()...** check boxes.

The first check box causes OutputDebugStringChecker to report function usage that is controlled by conditionals that are specified in the settings.

The second check box causes OutputDebugStringChecker to report function usage that is not controlled by conditionals that are specified in the settings.

You can also highlight the use of functions that are not controlled by conditionals that are specified in the settings by using the **Highlight __debugbreak()...** check box.

Reset
Click **Reset** to reset all settings on all pages of the settings dialog.

7.2 Functions

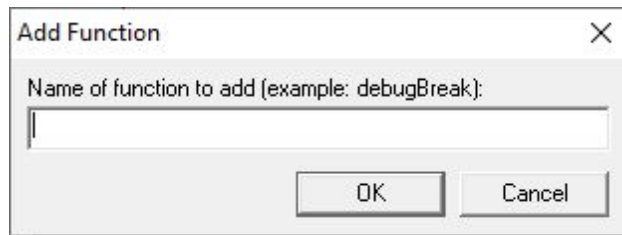The Finctions page of the settings dialog allows you to specify functions Debug Break Checker will search for.

The default list of functions that Debug Break Checker will scan for are listed here.

You can add and remove functions that should be scanned.

Add
Click **Add** to add a new function.

The Add Function dialog box will be displayed. Type the name of the new function and click OK.

Remove
Click **Remove** to remove the selected functions.

Remove All
Click **Remove All** to remove all functions.

Defaults
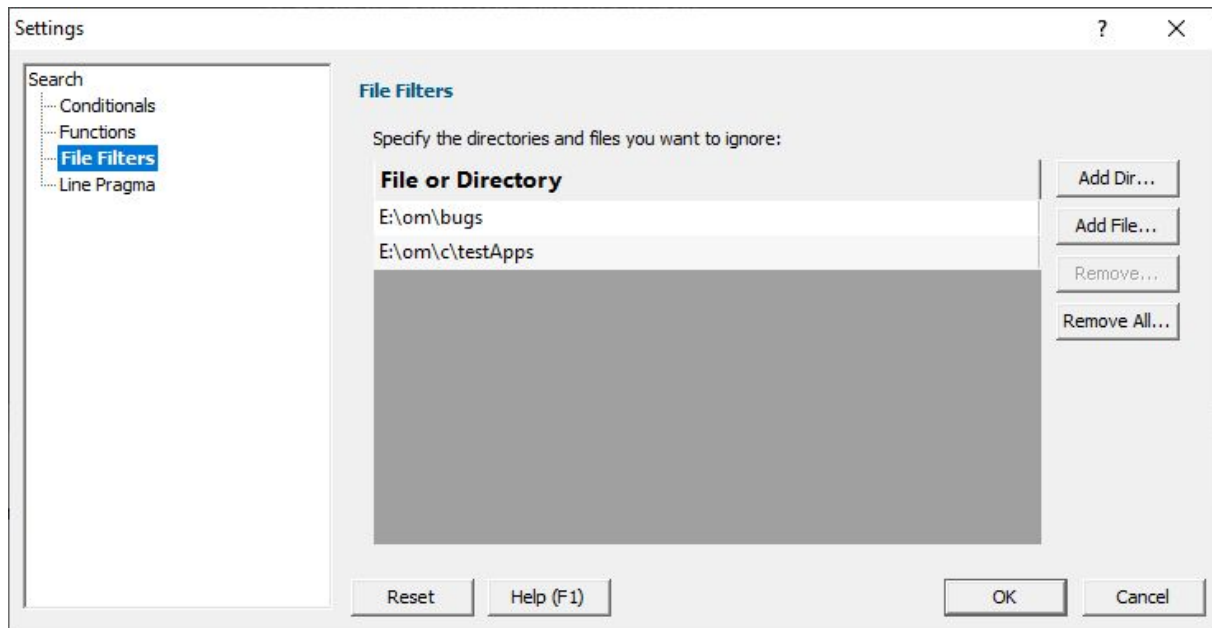Click **Defaults** to restore the default functions.

Reset
Click **Reset** to reset all settings on all pages of the settings dialog.

7.3   File Filters

The File Filters page of the settings dialog allows you to specify directories and files that Debug Break Checker will ignore when searching for functions.

This allows you to ignore directories files where you deliberately have special case usage of these functions (unit tests, development work, custom use cases) that would trigger warning results in Debug Break Checker that you don't want.

Add Dir...
Click **Add Dir...** to add a directory. The Microsoft directory chooser will be displayed.

Add File...
Click **Add File...** to add a file. The Microsoft file chooser will be displayed.

Remove
Click **Remove** to remove the selected directories and files.

Remove All
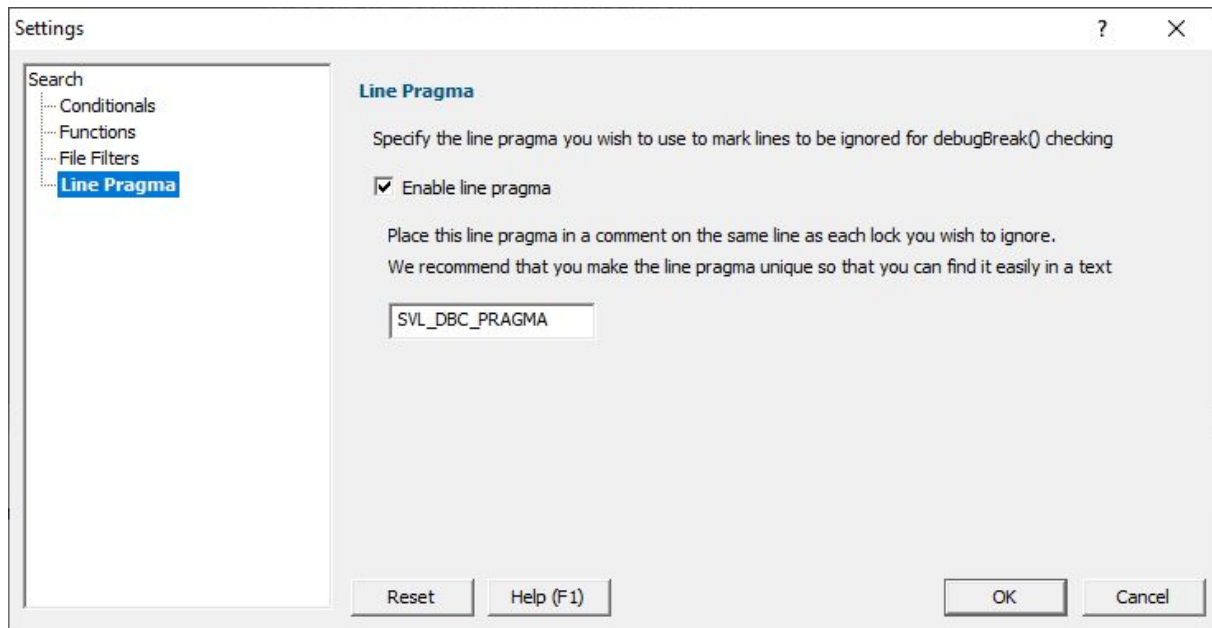Click **Remove All** to remove all directories and files.

Reset
Click **Reset** to reset all settings on all pages of the settings dialog.

7.4   Line Pragma

The Line Pragma page of the settings dialog allows you to specify a line pragma that can be used to mark individual locks to be ignored when checking for function call usage.

This allows you identify special use cases in your source code to avoid FALSE positives from Debug Break Checker.

Settings — Line Pragma dialog:

Line Pragma

Specify the line pragma you wish to use to mark lines to be ignored for debugBreak() checking

☑ Enable line pragma

Place this line pragma in a comment on the same line as each lock you wish to ignore.
We recommend that you make the line pragma unique so that you can find it easily in a text

SVL_DBC_PRAGMA

Search
- Conditionals
- Functions
- File Filters
- **Line Pragma**

[Reset] [Help (F1)] [OK] [Cancel]

When line pragmas are enabled any line containing the specified line pragma in a comment (single line comment or multi-line comment) will be ignored when checking for function call usage.

If the line pragma is enabled, the line pragma cannot be empty or whitespace. We recommend a value that is unique across your codebase so that you can find all line pragmas easily if you need to search for them.

Example usage of the pragma in code:

```
__debugbreak(_T("NVIDIA Properties Dump\r\n"));       // SVL_DBC_PRAGMA
```

Part VIII

8   How to use Debug Break Checker

To use Debug Break Checker follow these basic steps:

1.   If you want to change the functions to scan for, **Edit > Settings...** to open the settings dialog.

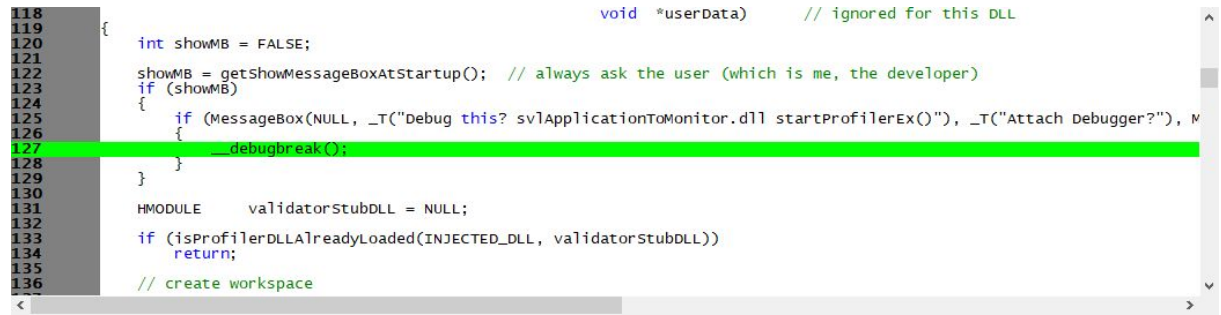2.   **File > Scan directory...** to display the directory chooser.

You can type the directory path or use the Microsoft directory chooser by clicking **Browse...**.

When you are happy with your choice of directory click **Start Scan** to start the process of finding the specified functions.

3.   If at any time you want to stop the scan **File > Stop scan**.

4.   Any function usages not filtered are shown in the grid.

5.   Select an entry shown in the grid to view the source code in the source code window.

Double click on the entry to edit the source code in Visual Studio.

6.   If you want to clear the results, **File > Clear**.

7.   If you want to filter a location by directory, right click on the entry in the list and choose **Filter Directory**.

8.   If you want to filter a location by filename, right click on the entry in the list and choose **Filter Filename**.

9.   If you want to mark a function to be ignored, right click on the entry in the list and choose **Add Line Pragma to Source Code**.

10.   If you want to go to the source directory, right click on the entry in the list and choose **Open Folder...**.

Part IX

9   Command Line Interface

Debug Break Checker can be used from the command line as well as with the GUI.

User interface
If command line arguments are detected the user interface is not displayed.

Error reporting
When running from the command line any errors are written to the output file specified on the command line.

Return code
The process return code is 0 for no function calls found.

If there are function calls found the return code is a positive integer describing the number of function calls found.

9.1   Alphabetic Reference

/addFileFilter

Adds a filename or directory to the list of file filters Debug Break Checker will examine when checking for function call usage.

/addFileFilter filename
/addFileFilter directory

Example: /addFileFilter e:\om\c\threadLockChecker\tlcExample\tlcExample.cpp
Example: /addFileFilter e:\om\c\threadLockChecker\tlcExample\

/addFunction

Adds a function to the list of functions Debug Break Checker will check.

/addFunction functionName

Example: ./addFunction myDebugInformationFunction

/dir

Specifies the directory hierarchy that will be scanned by Debug Break Checker.

/dir directory

Example: /dir e:\om\c\

/linePragma

Specifies the line pragma to use if line pragmas are enabled.

/linePragma pragma

Example: /linePragma SVL_DBC_PRAGMA

Example usage of the pragma in code:

```
__debugbreak();                    // SVL_DBC_PRAGMA
```

/linePragmaEnable

Enables or disables the ignoring of locks if a line pragma is found in a comment on the same line as the function call.

If you use this option to enable line pragmas you must use **/linePragma** to specify a line pragma to use. If you don't specify a line pragma this option does nothing.

/linePragmaEnable:On|Off

Example: /linePragmaEnable:On
Example: /linePragmaEnable:Off

/loadSettings

Specifies a file to load that contains settings that will override the current settings for this command line search.

/loadSettings filename

Example: /loadSettings e:\tests\test1\testSettings.tlc

/output

Specifies a filename to write the results of the command line search.

If no errors are found any file with this name will be deleted.

If errors are found a text file will be written containing error information. Error information will be listed one error per line.

/output filename

Example: /output e:\tests\test1\testResults.txt

/removeAllConditionals

Removes all conditionals from the list of conditionals Debug Break Checker will use to check when checking for function call usage.

This option should be used before any use of /addConditional.

/removeAllConditionals

/removeAllFileFilters

Removes all file filters from the list of file filters Debug Break Checker will use to check which files and directories to ignore when checking for function call usage.

This option should be used before any use of /addFileFilter.

/removeAllFileFilters

/removeAllFunctions

Removes all functions from the list of functions Debug Break Checker will check.

This option should be used before any use of /addFunction.

/removeAllFunctions

/reportConditional

Debug Break Checker will only report functions that are conditionally compiled inside a define specified by the settings

/reportConditional:On|Off

Example: /reportConditional:On
Example: /reportConditional:Off

/reportNonConditional

Debug Break Checker will only report functions that are not conditionally compiled inside a define specified by the settings

/reportNonConditional:On|Off

Example: /reportNonConditional:On
Example: /reportNonConditional:Off

/reset
Reset all settings to the default values for this command line search.

You should put this first on your command line to create a known default state prior to setting values using the command line.

/reset

9.2   Usage Reference

Search

/dir
Specifies the directory hierarchy that will be scanned by Debug Break Checker.

/dir directory

Example: /dir e:\om\c\

Report

/reportConditional

Debug Break Checker will only report functions that are conditionally compiled inside a define specified by the settings

/reportConditional:On|Off

Example: /reportConditional:On
Example: /reportConditional:Off

/reportNonConditional

Debug Break Checker will only report functions that are not conditionally compiled inside a define specified by the settings

/reportNonConditional:On|Off

Example: /reportNonConditional:On
Example: /reportNonConditional:Off

Conditionals

/addConditional

Adds a conditional to the list of conditionals Debug Break Checker will check.

/addConditional conditional

Example: ./addConditional _DEBUG

/removeAllConditionals

Removes all conditionals from the list of conditionals Debug Break Checker will use to check when checking for function call usage.

This option should be used before any use of /addConditional.

/removeAllConditionals

Functions

/addFunction

Adds a function to the list of functions Debug Break Checker will check.

/addFunction functionName

Example: ./addFunction myDebuggingFunction

/removeAllfunctionss

Removes all functions from the list of functions Debug Break Checker will check.

This option should be used before any use of /addFunction.

/removeAllFunctions

File Filters

/addFileFilter

Adds a filename or directory to the list of file filters Debug Break Checker will examine when checking for function call usage.

/addFileFilter filename
/addFileFilter directory

Example: /addFileFilter e:\om\c\threadLockChecker\tlcExample\tlcExample.cpp
Example: /addFileFilter e:\om\c\threadLockChecker\tlcExample\

/removeAllFileFilters

Removes all file filters from the list of file filters Debug Break Checker will use to check which files and directories to ignore when checking for function call usage.

This option should be used before any use of /addFileFilter.

/removeAllFileFilters

Line Pragma

/linePragmaEnable

Enables or disables the ignoring of locks if a line pragma is found in a comment on the same line as the function call.

If you use this option to enable line pragmas you must use **/linePragma** to specify a line pragma to use. If you don't specify a line pragma this option does nothing.

/linePragmaEnable:On|Off

Example: /linePragmaEnable:On
Example: /linePragmaEnable:Off

/linePragma

Specifies the line pragma to use if line pragmas are enabled.

/linePragma pragma

Example: /linePragma SVL_TLC_PRAGMA

Example usage of the pragma in code:

```
__debugbreak();                    // SVL_DBC_PRAGMA
```

Miscellaneous

/loadSettings

Specifies a file to load that contains settings that will override the current settings for this command line search.

/loadSettings filename

Example: /loadSettings e:\tests\test1\testSettings.tlc

/output

Specifies a filename to write the results of the command line search.

If no errors are found any file with this name will be deleted.

If errors are found a text file will be written containing error information. Error information will be listed one error per line.

/output filename

Example: /output e:\tests\test1\testResults.txt

/reset

Reset all settings to the default values for this command line search.

You should put this first on your command line to create a known default state prior to setting values using the command line.

/reset

9.3   Command Line Examples

This section provides some example command lines. For each example we'll break it down, argument by argument so that you can see how the command line works.

In all these examples the executable to run is **debugBreakChecker.exe**. You will need to provide the full path to debugBreakChecker.exe, or add the path to debugBreakChecker install directory to your $PATH.

Example 1

debugBreakChecker.exe /dir e:\om\c\ /output e:\test.txt

/dir e:\om\c

Scan the directory e:\om\c\

/output e:\test.txt

Write the output to e:\test.txt. If there are no functions detected there will be no output file.

Example 2

debugBreakChecker.exe /dir e:\om\c\ /output e:\test.txt /removeAllFunctions /addFunction myDebuggingFunction

/dir e:\om\c

Scan the directory e:\om\c\

/output e:\test.txt

Write the output to e:\test.txt. If there are no functions detected there will be no output file.

/removeAllFunctions

Remove all existing functions definitions.

/addFunction myDebuggingFunction

Add one function definition so that Debug Break Checker is checking for just the function **myDebuggingFunction**

Example 3

debugBreakChecker.exe /loadSettings e:\tests\test1\settings.tlc /output e:\test.txt

/loadSettings e:\tests\test1\settings.tlc

Load settings from the file e:\tests\test1\settings.tlc.

/output e:\test.txt

Write the output to e:\test.txt. If there are no errors there will be no output file.

Example 4

debugBreakChecker.exe /loadSettings e:\tests\test1\settings.tlc /output e:\test.txt /dir e:\om\c

/loadSettings e:\tests\test1\settings.tlc

Load settings from the file e:\tests\test1\settings.tlc.

/output e:\test.txt

Write the output to e:\test.txt. If there are no errors there will be no output file.

/dir e:\om\c

Scan the directory e:\om\c\