User Guide

Test coverage

Department:	Aalborg Wireless Center
Creation Date:	18 December, 2002
Last Modified:	18 June, 2003 by Carina Graversen
ID and Version:	8434.521.03.005
Status:	Accepted

0 Document Control

0.1 Document History

ID	Author	Date	Status
8434.521.03.001	KSP	18 December, 2002	Being Processed
8434.521.03.002	KSP	5 February, 2003	Being Processed
8434.521.03.003	KSP	6 February, 2003	Accepted
8434.521.03.004	CGR	17 June, 2003	Accepted
8434.521.03.005	CGR	18 June, 2003	Accepted

0.2 References, Abbreviations, Terms

[TI 8010.801] 8010.801, References and Vocabulary, Texas Instruments

Table of Contents

1 Introduction

If you are already familiar with profiling you can jump to chapter 3 for a quick guide on how to perform the measurements and skip the rest of the document.

Otherwise you might want to start reading about how to set up the environment to perform the profiling, which is described in chapter 2. This includes both setting up Visual Studio, the TAPCALLER and defining which source files are to be profiled.

This user guide will concentrate on performing the following profiling methods:

- Function profiling
    - Function timing
    - Function coverage
- Line profiling
    - Line coverage

This small user guide will briefly explain how to perform the different measurements mentioned above and give an example of how to measure test coverage on an existing entity. Executing some different batch files will carry out this profiling. The principal line of the measurement process consists of a setup batch file, a runtime batch file and a post processing batch file.

2 Setting up environments for both GPRS and UMTS

This chapter contains a small guide to set up the environment correctly.

1. The following lines must be inserted in the ConfigSpec[1]:
   ```
   element /gpf/DOC/…TEST_COVERAGE
   element /gpf/BIN/…TEST_COVERAGE
   ```
   Below the line
   ```
   element * CHECKEDOUT
   ```

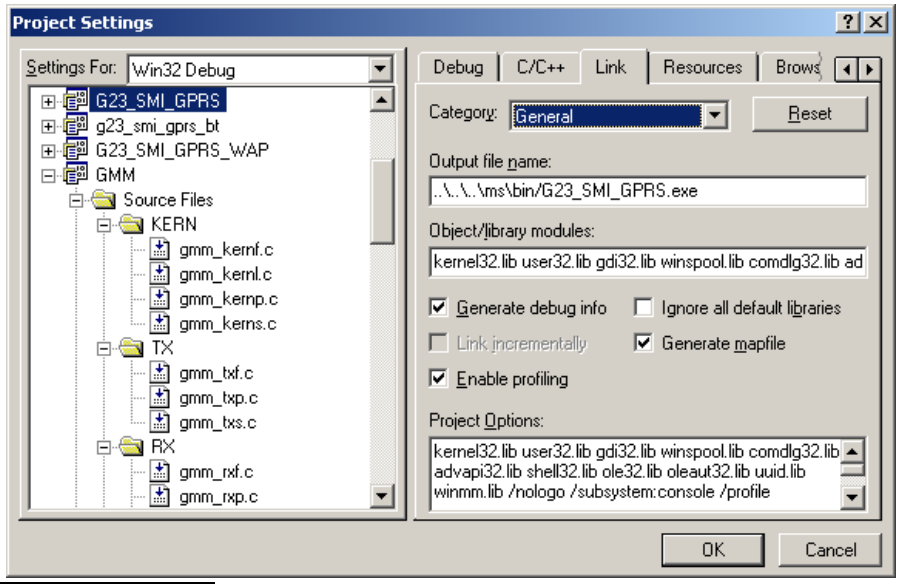
2. For GPRS, you must set up your system as described in x:\sw\2g\stability\G23M build env setup.doc

   It should be noted, that the path C:\Program Files\Microsoft Visual Studio\VC98\bin should be listed first in the environment path. This can be done by executing the following command:
   ```
   C:\Program Files\Microsoft Visual Studio\VC98\vcvars32.bat
   ```

3. Start a 4NT prompt and run the following command (z is the test view in this example):
   GPRS: `[z:\gpf]initvars gprs MS z: \g23m\Condat`
   UMTS: `[z:\gpf]initumts`

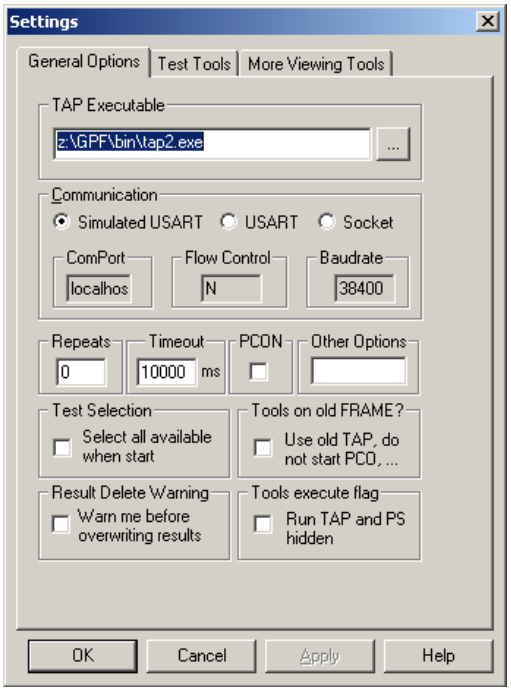
4. Compile the test cases from either Visual Studio or from a 4NT prompt this way (SNDCP used as an example)
   ```
   [z:\gpf\bin]mkalltc -f sndcp
   ```

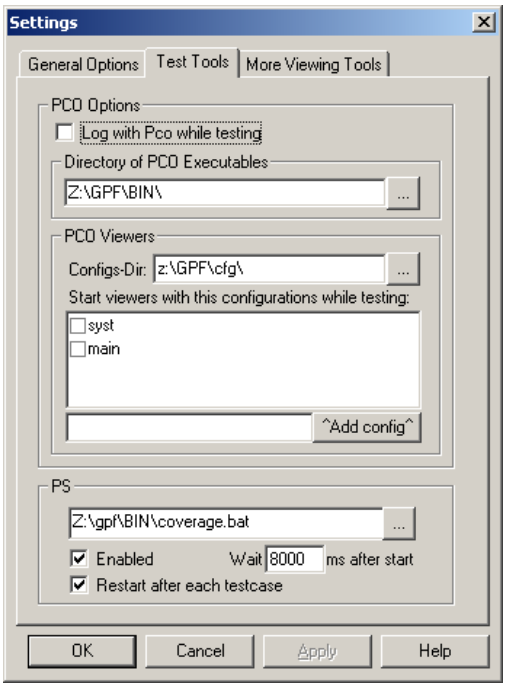
5. Enable profiling in Visual Studio 6. This is done by selecting the project properties: “Project” | “Settings” | ”link tab”. Make sure that “Enable profiling” and “Generate .map file” both are checked. This is illustrated below. Please note that it is necessary to rebuild the stack after changing these options for the profiling to work. Note that these settings can be leaved in for normal use as well.

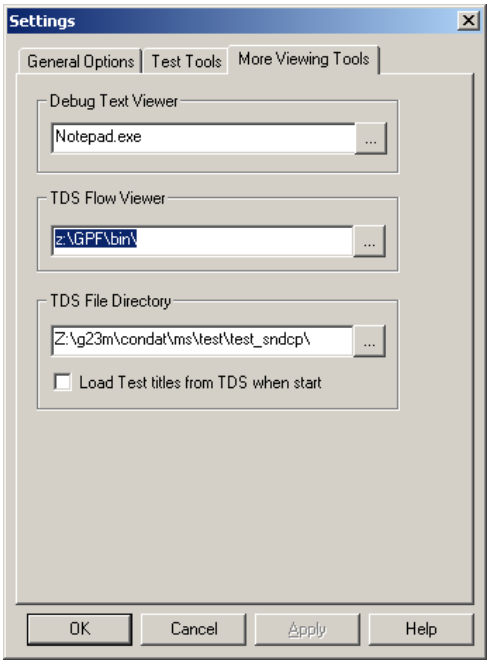


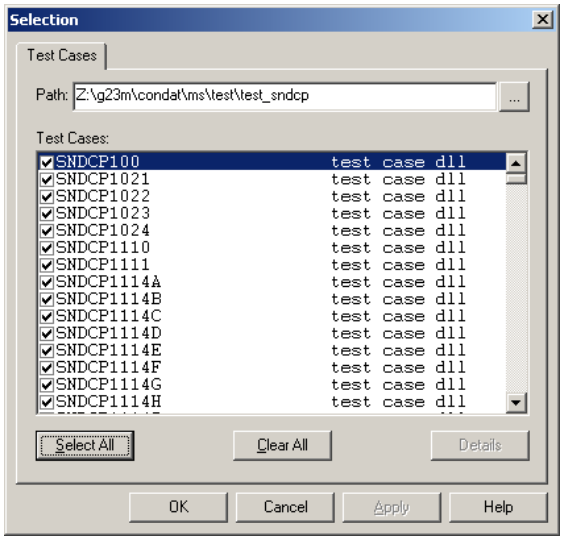
[1] This is only necessary until the Test Coverage functionality has been released as a part of the TOOL release. As the current version is not part of a tool release it can be subject to changes.

6. Start TAPCALLER from directory z:\gpf\bin, and set up the TAPCALLER as shown below (z is testview in this example, and SNDCP is the test entity)[2]:









7. In order to perform the profiling it is necessary to specify the desired source files and the executable to be profiled. This information could be passed on as arguments when executing the profiling job. However this is not very feasible, as it would result in a very long and complicated argument list.

[2] In case of test coverage for UMTS, the PCON field should be checked in the settings ->General options, as well as "other options" should contain the string »–v«. Run TAP and PS hidden might also be checked.

Instead specifying the source code to be measured in a specific list file does this. Such a list file should exist for each entity so it can be used from time to time. In addition to this there can be several different list files for each entity for different measurements.

The list files should be saved in a simple text file with suffix .rsp such as for instance sndcp.rsp. This way it will be fairly simple to edit the arguments as it can be done in any text editor and it will be possible to reuse the information and hence it will be easy to run several profiling measurement under the same conditions. Following the result can be compared directly.

The source files should be included one by one with one source file per line. In front of the file name an /INC or /EXC should be included indicating whether the file shall be included or not. This way it is possible to add future files to the list without including them in the current measurement. In addition to this it is also possible to use comments by use of an #.

This will e.g. result in a file like the following:

```
/INC sndcp_mgf.c(0-0)    # this is a comment
/INC sndcp_mgp.c(0-0)
/EXC sndcp_mgs.c(0-0) #This shall not be included for now.
```

The (0-0) indicates that all lines in the file will be profiled. Other combinations such as (10-20) can also be used and would result in line 10 to 20 will be profiled. When performing function coverage or function timing measurements it is necessary to specify the object file for the source files in question.. Therefore it might be a good idea to have a list of source files for line coverage measurements and one for function measurements. An example of an rsp file for function measurements could be:

```
/INC sndcp_mgf.obj    # this is a comment
/INC sndcp_mgp.obj
/EXC sndcp_mgs.obj #This shall not be included for now.
```

The created rsp list files should be placed together with the source files for the test cases. This means that for an UMTS entity, the list files should be placed under \g23m\Condat\ms\src\sm\test_usm\. For a GPRS entity, the list files should be placed under \g23m\Condat\ms\doc\test. This is also where the result files will be placed.

The system has now been set correctly to perform the measurements, which are described in chapter 3.

3 Profiling

When the system has been set up properly, it is possible to begin the actual profiling.

1. Setup the test environment (PCO).

2. Setup the desired profiling method by starting the appropriate batch file plus the name of the entity list file in question (This list file can either be an existing file or a new one can be created). In addition to this it is necessary to provide the directory path to the entity test files starting from the \g23m\condat\ms\src\ directory for UMTS and \g23m\condat\ms\ for GPRS. That is, for the UMTS SNDCP entity this parameter would be sndcp\test_usm\ while for the GPRS SNDCP entity it would be doc\test\. The last required parameter is either UMTS or GPRS. The parameters have to be used in the order given.

   It is possible to perform 3 different kinds of measurements. With sndcp.rsp file as argument these three options are available:

   GPRS: `[z:\gpf\bin]setup_line_coverage.bat sndcp.rsp doc\test\ gprs`
   GPRS: `[z:\gpf\bin]setup_function_coverage.bat sndcp. rsp doc\test\ gprs`
   GPRS: `[z:\gpf\bin]setup_function_timing.bat sndcp. rsp doc\test\ gprs`

   UMTS: `[z:\gpf\bin]setup_line_coverage.bat sndcp. rsp sndcp\test_usm\ umts`
   UMTS: `[z:\gpf\bin]setup_function_coverage.bat sndcp. rsp sndcp\test_usm\ umts`
   UMTS: `[z:\gpf\bin]setup_function_timing.bat sndcp. rsp sndcp\test_usm\ umts`

3. Decide whether the PS should be restarted for each test case.
   a. If it should be restarted for each case: set up the TAPCALLER accordingly (“Configuration” | “Settings” | “Test Tools”, check “Restart after each test case”).
   b. If it should be started only once: Use TAPCALLER with “Restart after each test case” disabled or start the coverage by executing the batch job coverage.bat from the 4NT prompt.

   When choosing whether the protocol stack should be restarted after each test case or not, it must be considered if several testcases can be passed without restarting the stack. If this is the case, it is recommended not to restart the protocol stack, since this will eliminate the overall test time.

4. Execute the desired test cases from the TAPCALLER.

   It should be noted, that if a test case results in the protocol stack to be hanging, it is very important to close the protocol stack and not the other window! Preferably the post_processing.bat should be used for this.

5. When all testcases have been run, finish the profiling process by executing the post processing batch file post_processing.bat from the 4nt prompt.

   ```
   [z:\gpf\bin]post_processing
   ```

6. The profiling has now finished and the results can be found in the same directory as the .rsp file. The results for e.g. line coverage are saved in the following files: “exe-name”-LV.lst, “exe-name”-LV2.lst, “exe-name”-LV3.lst and in a file called “exe-name”-LV.out. For respectively function coverage and function timing the suffix would be FV or FT instead of LV. The first file contains the

statistics such as lines of code and percentage run while the second only is used for generating the last output file for the Visual Studio debugger (this is currently only applicable for the line coverage method). The LV3-file however contains all lines of the sourcecode, where a "*" in front of the line indicates that this line has been touched, which is not the case if a "." is put in front of the line. Se example below:

```
*       dti_channel->erase_channel = FALSE;
.       used_dti_channels &= ~tst_id;
```

Line touched
Line not touched

7. In case the test results should be used in a report, the result from the TAPCALLER can be written to a text-file, which will be easy to paste into a Word-document. The text-file is created this way:

   In the menu "Files" | "Save as" in the TAPCALLER the result of the test is saved.

   Go to the 4NT prompt, and go to the directory where the file was saved, and write (without the "-characters):

   ```
   "ClearCaseDirectoryLetter":\UMTS\Tool_Documents\Utils\tapcaller_prettyprint.exe "Name of the file you saved" > report.txt
   ```

   The result can now be found in the same directory as the file from the TAPCALLER was saved.

   Please note that the ClearCaseDirectoryLetter must be an UMTS view.

4 Profiling example

This section will demonstrate a short example of how to perform profiling on some specific code for UMTS. In this example the entity SM is chosen. As described earlier, before performing profiling for the first time it is necessary to setup Visual Studio to generate profiling info. This is done by selecting the project properties: "Project" | "Settings" | "link tab". Make sure that "Enable profiling" and "Generate .map file" both are checked. This is illustrated in Figure 1. Please note that it is necessary to rebuild the stack after changing these options for the profiling to work. (These settings can be leaved in for normal usage as well.)

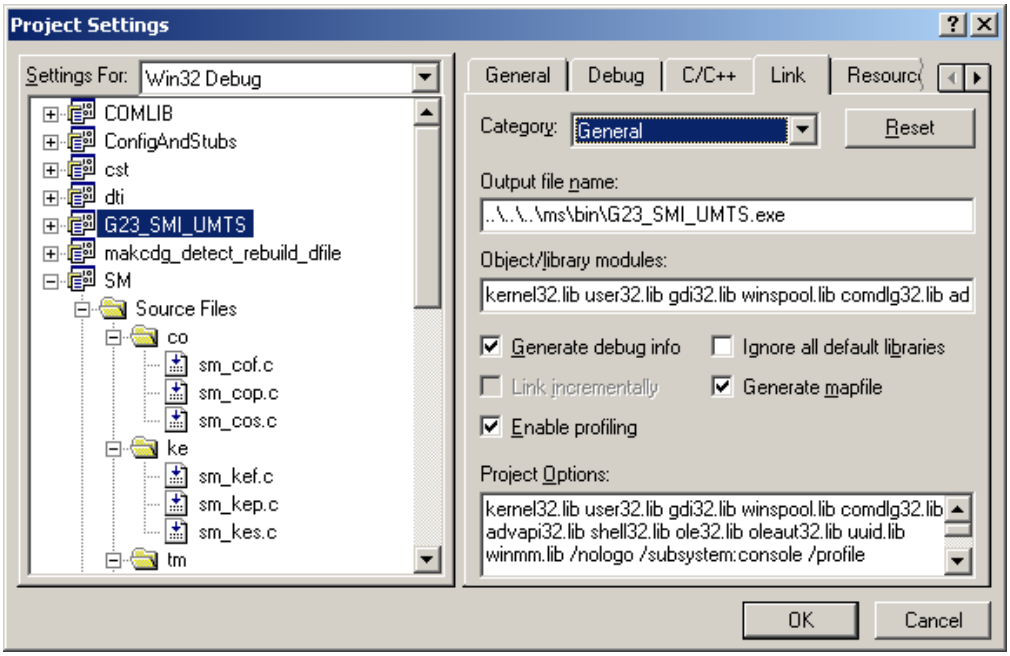


Figure 1: Enabling profiling in Visual Studio

After enabling the profiling make sure to rebuild the project. Otherwise the profiling will not work. Naturally the test cases also have to be generated if not already present (not necessary to rebuild).

In order to specify which source code is to be measured upon it is necessary to provide this as a file argument to the setup. The list file used in this example is called sm.rsp and it looks as follows:

```
/INC sm_cof.c(0-0)
/INC sm_cop.c(0-0)
/INC sm_cos.c(0-0)
/INC sm_kef.c(0-0)
/INC sm_kep.c(0-0)
/INC sm_kes.c(0-0)
/INC sm_tmf.c(0-0)
/INC sm_tmp.c(0-0)
/INC sm_tms.c(0-0)
/INC sm_f.c(0-0)
/INC sm_pei.c(0-0)
/INC sm_qos.c(0-0)
/INC sm_tft.c(0-0)
```

This example will use the method where the stack is restarted for each test case as this is the most complicated of the two. The TAPCALLER is setup with this option checked. The delay used in this example has been set to 8000 ms. This is illustrated in Figure 2

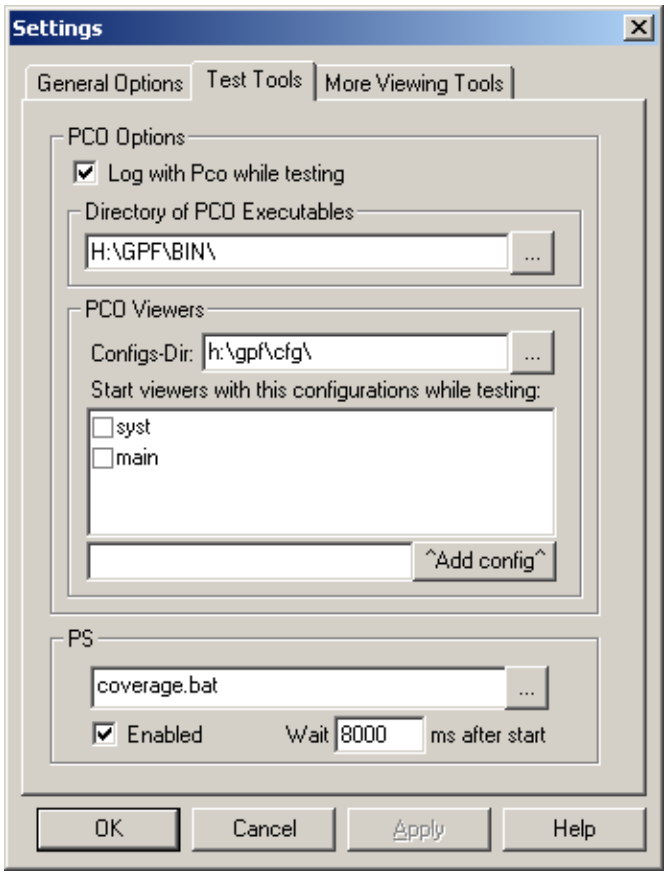


Figure 2: The setting used for this example.

In order to start the actual measurements the setup_line_coverage.bat is executed as follows:

setup_line_coverage.bat sm.rsp sm\test_usm\

Following all the desired test cases should be executed from the TAPCALLER. After the test case execution has finished the post_processing.bat is executed.

4.1 Viewing the results

The results can now be found in 2 output files: G23_SMI_UMTS-LV.lst and G23_SMI_UMTS-LV.out. In the first file the percentage of covered code can be found. This is illustrated in the following figure.

```
    Command line at 2003 Jan 24 09:59: G23_SMI_UMTS
    Command line at 2003 Jan 24 09:59: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:00: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:00: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:00: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:01: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:01: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:01: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:01: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:01: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:02: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:02: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:02: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:02: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:03: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:03: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:03: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:03: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:03: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:04: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:04: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:04: G23_SMI_UMTS
    Command line at 2003 Jan 24 10:04: G23_SMI_UMTS
    Total lines: 2678
    Line coverage: 64,7%

Module Statistics for g23_smi_umts.exe
---------------------------------------
    Lines in module: 2678
    Module line coverage: 64,7%

 Hit
Count    %   Line
-----------------
   35   0,1 (sm_f.c:124)
   35   0,1 (sm_f.c:126)
   35   0,1 (sm_f.c:129)
```

Figure 3: The output from the line coverage.

The latter can be added to the Visual Studio project. By changing the properties as depicted in Figure 4.

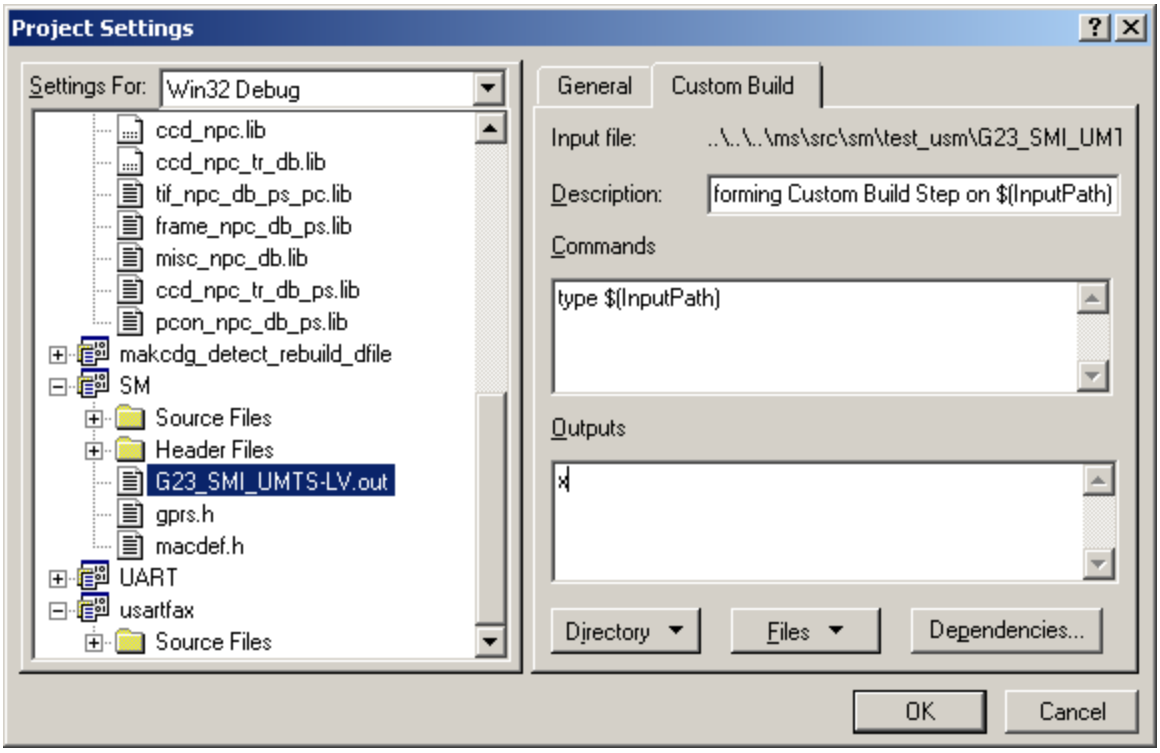


Figure 4: Adding the .out file to the project and changing its properties

Following the file can be compiled and will be typed in the debug window. This is illustrated in Figure 5.

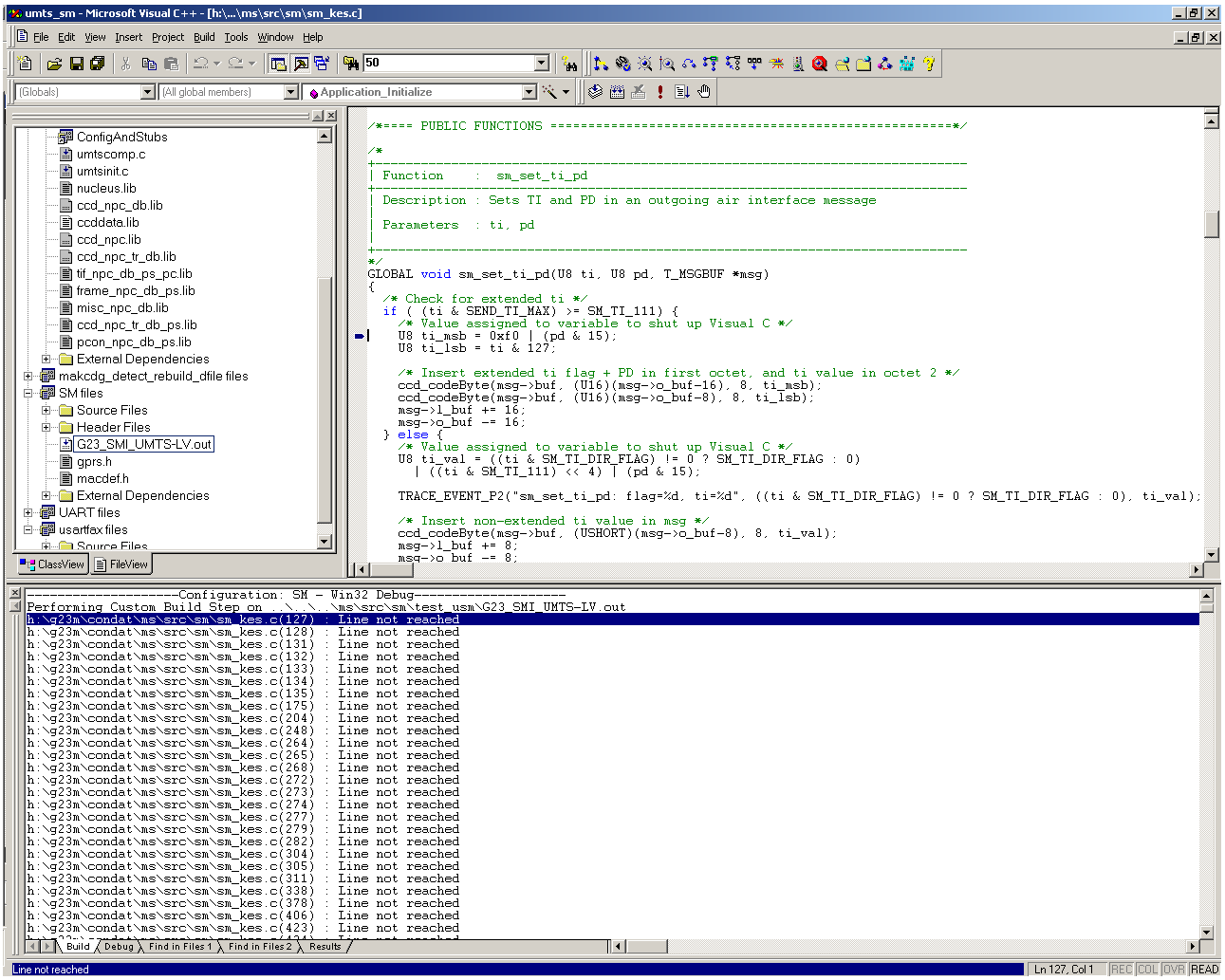


Figure 5: The out file can now be used for easy access to the lines in question.

This concludes the example of a line coverage measurement. The procedure for performing function coverage and function timing are like for line coverage but merely requires that another setup batch file is run. The only real difference is that it is necessary to specify the .obj files instead of the .c files.

Currently the feature to double click on a line in Visual Studio for easy access is only supported for line coverage.

5 Known bugs

The test coverage functionality contains a few known bugs, which will be corrected in the future.

The known bugs are:

- **In old versions of the 4NT prompt, the command line might get too long in case of many sourcefiles. Solution is to use version 4 or higher of the 4NT prompt.**