User's Guide
SPNU636–January 2020

Hercules™ Software Diagnostic Library CSP Without LDRA

Contents

1 Introduction 2
2 Software Requirements 2
3 Software Diagnostic Library TAU Tool Restrictions 2
4 Terminologies Used in Software Diagnostic Library TAU 2
5 Functional Blocks of Software Diagnostic Library TAU 4
6 Software Diagnostic Library TAU Test Flow 5
7 Manual Settings to the LDRA Install Needed by the User 7
8 Steps for Using the Software Diagnostic Library TAU 10
9 Inputs to Software Diagnostic Library TAU 15
10 How to Add Individual Test Cases 19
11 Reports 21
12 FAQ 22

List of Figures

1 Excel Test Case Snapshot 3
2 Automated Static and Dynamic Analysis Flow 5
3 Test Automation Framework for CSP 6
4 Select the Compiler Options 7
5 Select Compiler 7
6 Open LDRAunit 8
7 Select Multiple Files 8
8 Delete the Set for the Device Under Test (Here RM46x Software Diagnostic Library is Selected Here) 9
9 LDRA Analysis Lock 9
10 Software Diagnostic Library GUI Open Page 10
11 Device Family Selection 11
12 Target and Build Option Selection 12
13 Test Case Selection and Run 13
14 Test Execution 14
15 Compiler Option Select 16
16 Linker Option Select 17
17 Runtime Library Selection 18
18 Test Sequence Template 19
19 Folder Structure 19
20 Test Case Example 1 20
21 Test Case Example 2 21

Trademarks

Hercules, Code Composer Studio are trademarks of Texas Instruments.
All other trademarks are the property of their respective owners.

1 Introduction

Hercules Software Diagnostic Library CSP contains Test Automation unit (Software Diagnostic Library TAU), which is a tool that helps the user generate dynamic coverage analysis reports and regression reports for the diagnostic application program interface (APIs) provided in the library to support ISO26262 and IEC61508 assessments.

TI provides a compliance support package (CSP) for the Software Diagnostic Library to help safety customers go through ISO26262 and IEC61508 assessments.

The Software Diagnostic Library TAU comes with unit test cases for all the modules supported by the Software Diagnostic Library for Hercules Family of devices and the necessary Test Infrastructure to run these test cases. The Software Diagnostic Library TAU also provides infrastructure for the Hercules customers to add their own test cases.

NOTE: Since all of the device families do not provide same Hardware features, it should be noted that the Test cases listed or created anew in TAU shall vary depending on the device selected. Customers are advised to see the device-specific Technical Reference Manuals and Data Sheets to ensure any new test cases to be added to TAU are valid for the target device.

2 Software Requirements

- OS: Windows version 7 or higher
- Software Diagnostic Library v02.02.00 (or higher)
- Perl 5.x. Download Link http://www.perl.org/get.html#win32
- Code Composer Studio™ 6.0 (or higher)
- Microsoft Office 2007 (or higher)
- LDRAunit-TI-Qual 9.4.3 (or higher)

NOTE: LDRA Unit is not provided as a part of the CSP download. To re-run the unit tests using the included Test automation unit software, customers are required to purchase and install LDRA Unit software from LDRA. This guide explains LDRA Unit installation settings required for the Test automation unit.

3 Software Diagnostic Library TAU Tool Restrictions

- This tool supports device families TMS570LS31x, TMS570LS21x, RM48x, TMS570LS12x, TMS570LS11x, RM46x, TMS570LS04x, TMS570LS03x, RM42x, TMS570LS09x, TMS570LS07x, RM44x, TMS570LCx, RM57x only.
- This tool does not support testing of assembly files.

4 Terminologies Used in Software Diagnostic Library TAU

4.1 What is Unit Testing?

In simple words, unit testing is a single function tested in isolation. Unit testing generally involves taking a subset of the software, linking it with a test driver and exercising it, and checking that the unit behaves as expected. This subset of the software could be anything from just a function to the entire software. The source file under test is instrumented and tested in white box mode to get the code coverage.

4.2 What is a Test Sequence?

A test sequence is a set of test cases, unit or functional (not both), targeted on a single c file. A test sequence is written in the form an Excel sheet listing the following:

- Global declarations
- Global code
- Function tested in each test case
- Input parameters for each test case
- Pass or fail criteria for each test case
- Variable declarations, startup code, and cleanup code for each test case

In addition for the traceability report generation, the following artifacts are also added for each test case:

- Test case ID
- Requirements covered by the test case

Each test sequence is converted to the TCF file, which is the actual input to the LDRA tool. Figure 1 shows an example of a test case in the sequence. See Section 10 for more information on how to write test cases.

			functional - get status of pbist - PASS if evaluation is TRUE,st_Result is ST_PASS		
		TestCase Description			
			DMA5A:4		
		TestCase ID & Require	SL_SR108, SL_SR109		
			failinfo.stResult = ST_FAIL;		
			while (1 !=		
			SL_SelfTest_Status_PBIST(&failinfo));		
		StartupCode			
5	..\..\..\safety_library\source\sl_selftest.c	Name	param1	%	failinfo.stResult
	SL_SelfTest_Status_PBIST	Decl_type	SL_PBIST_FailInfo*	boolean	SL_SelfTest_Result
		3 User_type	Input parameter applied at call	Function result	Output global
		Value	&failinfo	FALSE	ST_PASS
	Global Variables	File	..\..\..\safety_library\source\sl_selftest.c	..\..\..\safety_library\source\sl_selftest.c	..\..\..\safety_library\source\sl_selftest.c
		3 Variable Name	failinfo	adcconfig	adcpinstatus
		Variable type	SL_PBIST_FailInfo	SL_ADC_Config	SL_ADC_Pinstatus

Figure 1. Excel Test Case Snapshot

Each test case in a sequence does the following:

- Runs any initialization code
- Configures I/O variables
- Invokes a single function with specified arguments
- Captures any return values that are to be checked
- Captures the value of any I/O variables that are to be checked
- Runs any custom checks, such as checking execution time
- Saves the results
- Runs any cleanup code

4.3 What is a TCF?

The test case file (TCF) contains all of the information required to run or re-run the test cases. The sequences are converted into TCF. The TCF contains the tags for the test case ids and requirement ids, which help with traceability. The LDRA tool can group TCFs with regression reports and can be stored for regression verification. This information can either be saved with the source file through a software configuration management (SCM) system, or it can be used as an annotation. Requirements based testing documentation, including why particular values were chosen and tags to map to a requirement management system, can be added for storage. The TCFs can be re-run from the command line and in batch mode so that as the source code changes, module interfaces and outputs can be verified.

4.4 What is Code Coverage?

Code coverage is a measure used to describe the degree to which the source code of a program is tested by a particular test suite. A program with high code coverage has been more thoroughly tested and has a lower chance of containing software bugs than a program with low code coverage.

The Software Diagnostic Library TAU uses LDRA in the back to generate the following code-coverage criteria:

- Statement coverage
- Branch coverage
- MC and DC Coverage

4.5 What is Regression Report?

Regression report is the consolidated test results report generated by LDRA running the functional and unit tests selected through the Software Diagnostic Library TAU.

5 Functional Blocks of Software Diagnostic Library TAU

The functional blocks of the Software Diagnostic Library TAU are the following:

- LDRAunit-TI-Qual
  - Helps generate dynamic analysis reports
  - Interfaces to CCS debug server scripts
- CCS Debug Scripts
  - Helps load and execute the test codes
- TI Test Cases
  - Excel-based unit test cases (per module) that are supported in the Software Diagnostic Library
- TI Test Script Engine
  - Instruments the targeted C file(s) through LDRA
  - Generates TCF files, which invokes LDRA
  - Generated and is executable through an automatically generated make file
  - Helps in consolidating the code coverage report and regression report generated by LDRA
- Software Diagnostic Library TAU GUI
  - GUI to help the user choose the following:
    - Device family to test with
    - Test select (Software Diagnostic Library or TPS Driver—currently TPS Driver is not supported)
    - Build options based on the device
    - Target configuration based on the boards and the debugger
    - Update the status information of every test sequence selected

6 Software Diagnostic Library TAU Test Flow

Figure 2 and Figure 3 show the typical automated test flow followed by the Software Diagnostic Library TAU.

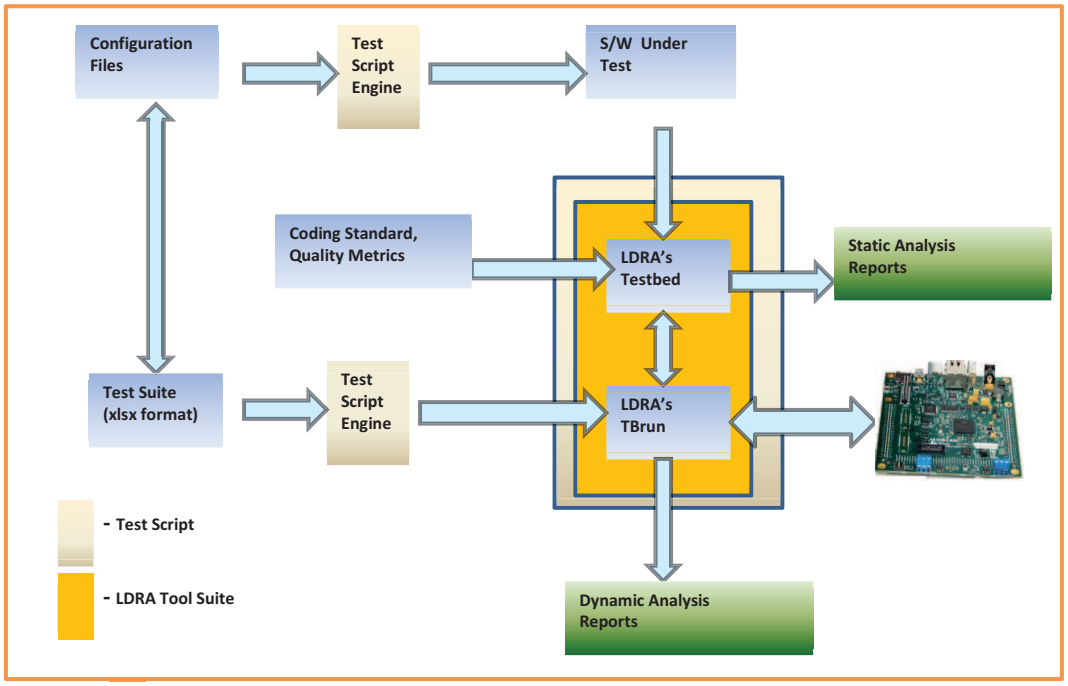


Figure 2. Automated Static and Dynamic Analysis Flow

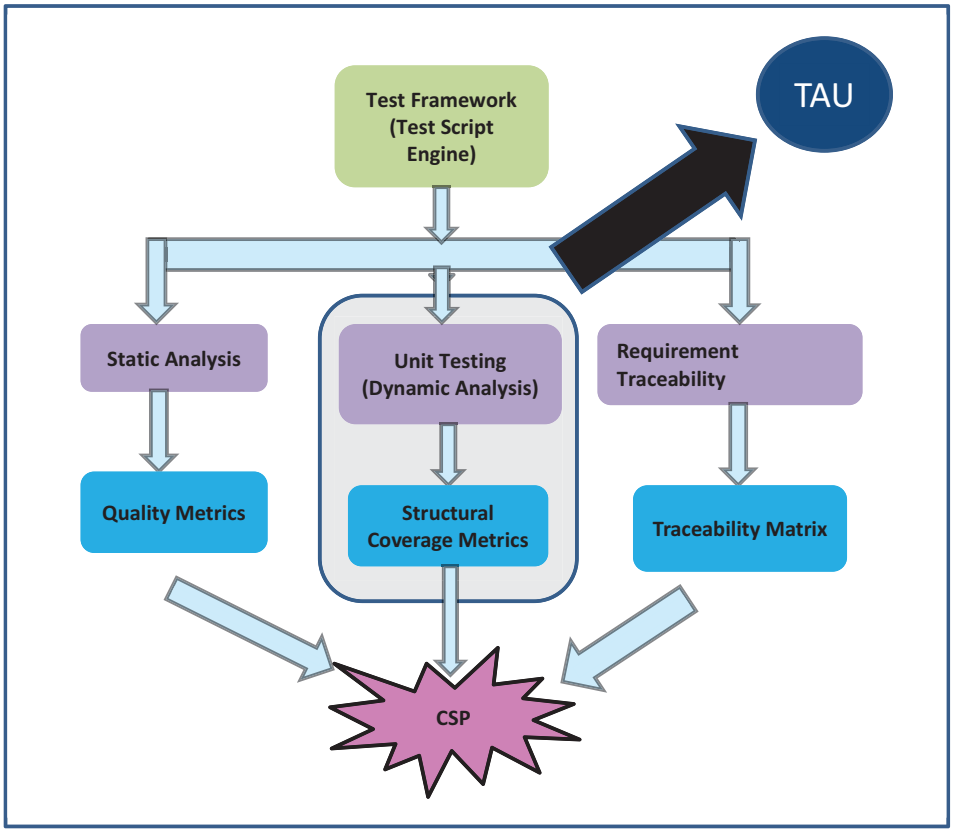


Figure 3. Test Automation Framework for CSP

7 Manual Settings to the LDRA Install Needed by the User

1. If the user has already installed LDRAunit-TI-Qual_C_CPP_9.4.3 for HALCogen, then the compiler option must be changed using the “compiler options” executable from the LDRAunit-TI-Qual_C_CPP_9.4.3 (run this program in administrative mode). Figure 4 and Figure 5 explain how to change the compiler options.

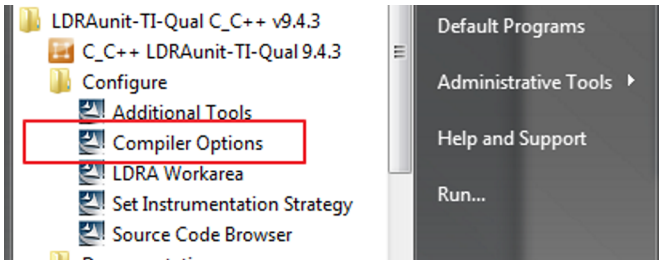


Figure 4. Select the Compiler Options

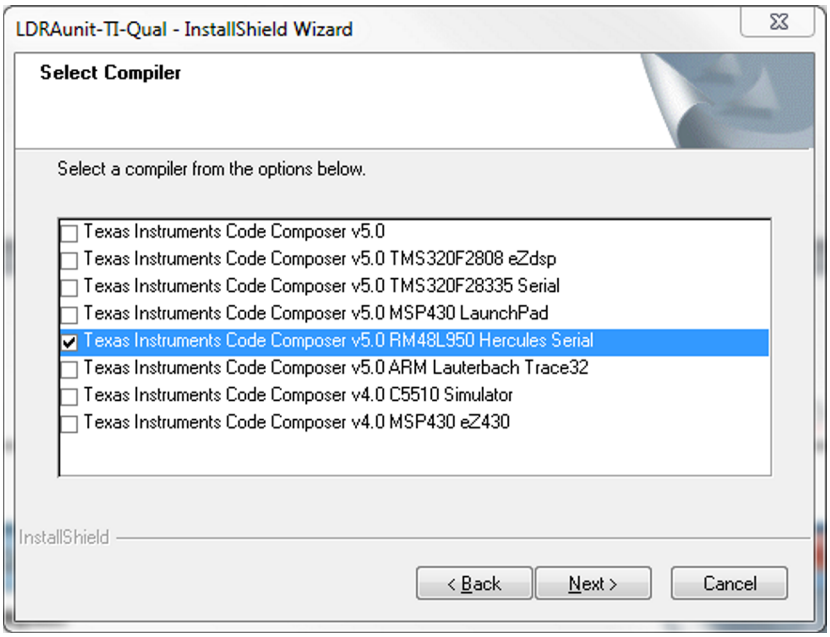


Figure 5. Select Compiler

2. Open the file LDRA_execute.bat under the LDRA installation directory (typically C:\Program Files (x86)\LDRA\LDRAunit-TI-Qual_C_CPP_9.4.3\Compiler_spec\Ticcs50\Rm48l950_hercules_serial\).
   - In the beginning of the file, find a line similar to the following:
     cd “C:\Program Files (x86)\LDRA\LDRAunit-TI-Qual_C_CPP_9.4.3\Utils\Comporter”
   - Change the above line to:
     “cd /d C:\Program Files (x86)\LDRA\LDRAunit-TI-Qual_C_CPP_9.4.3\Utils\Comporter”

NOTE: The software may require administrative privileges to change this file. Changing the file is necessary because the CD only works when trying to change the directory in the current working drive. If the TAU is installed in another drive other than C: drive, the CD fails to change the directory.

3. Connect the device board to the system (PC) for test, depending on the Target Board HW configuration, using either of the following:
   1. USB cable on the SCI port, as SCI is used for testing with XDS100/XDS110 USB emulator integrated on board
   2. USB-JTAG External Debugger XDS510 connected using the JTAG lines on board

   .
4. When selecting a HALCoGen project or modifying an existing project under demo_app\HALCoGen, take care to see that the SCI continue on suspend bit (bit 17 in SCI Global Control Register 1 (SCIGCR1) in the SCI module) is enabled.
5. If LDRAunit-TI-Qual_C_CPP_9.4.3 for HalCogen has already installed, then the compiler option must be changed using the compiler, which is executable from the LDRAunit-TI-Qual_C_CPP_9.4.3 file.
6. In cases where issues occur when building and running test cases, the user may have a corrupt LDRAunit-TI-Qual_C_CPP_9.4.3 work area. To fix the issue, delete the existing sets (an LDRA work item) as shown in Figure 6 through Figure 8, and then start running the test cases again.

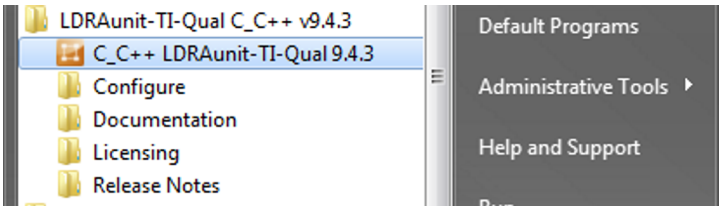


Figure 6. Open LDRAunit

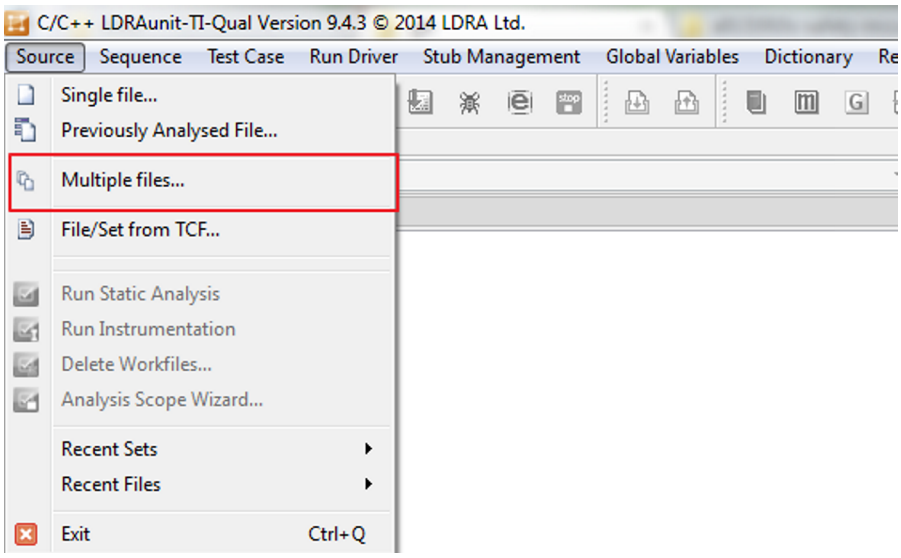


Figure 7. Select Multiple Files

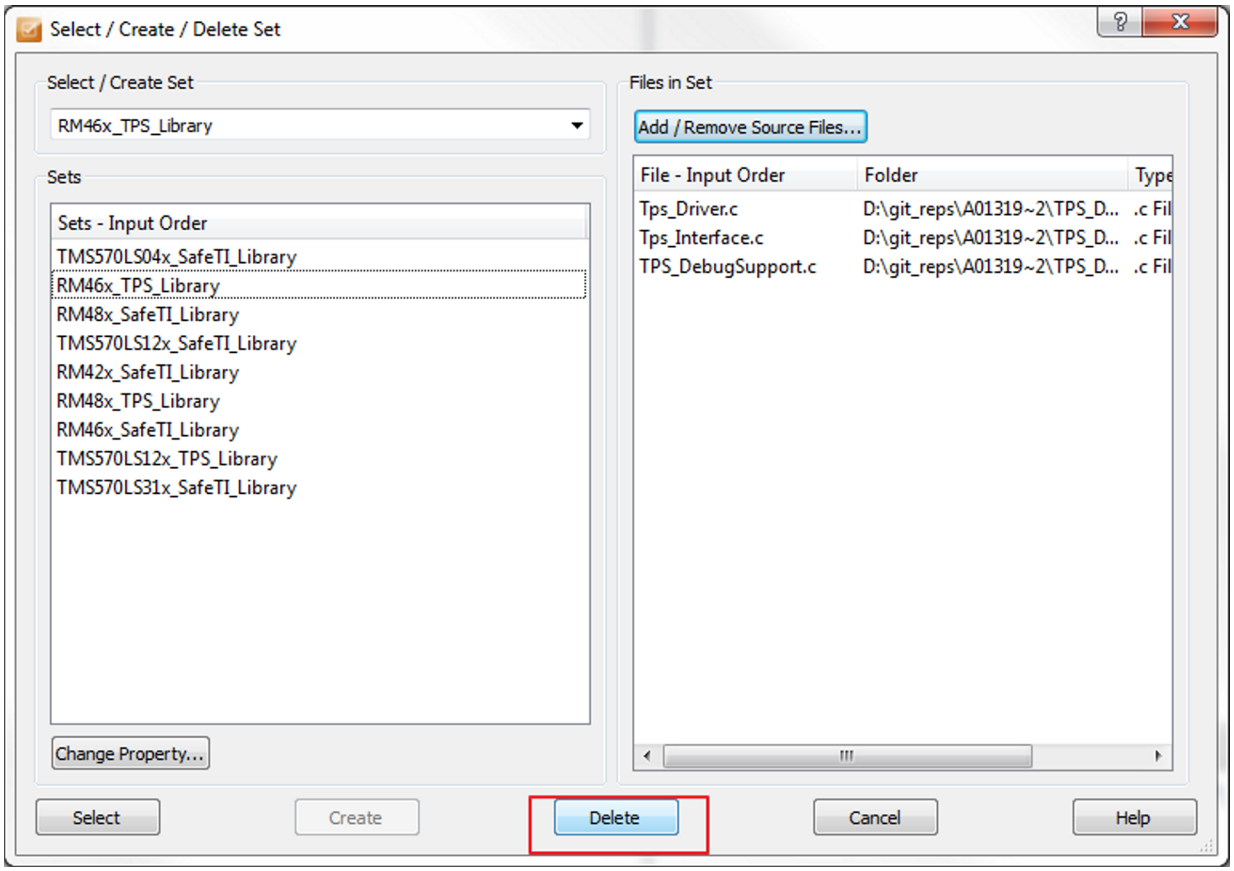


Figure 8. Delete the Set for the Device Under Test (Here RM46x Software Diagnostic Library is Selected Here)

7. Sometimes when the system hangs, the test execution is halted and the source code may be corrupted. That is, the user may leave the source code in an instrumented state (LDRA instrumentation). In that case, LDRA creates the source backup folder in: **<installation directory>\safety_library\source** Replace the corrupted source code using this backup.
8. The RTS libraries used for building the .out file for the test cases may sometimes be missing in the compiler. Automatic build of the RTS libraries may fail when the environment variables are not correctly set for the shell that is used for building the RTS library. The user must ensure that the required RTS libraries are available in the compiler.
9. In some cases, for example: forcibly closing test execution, ending the test automation task, or due to a power loss when the test execution is running, the LDRA analysis may lock (see Figure 9). This lock must be deleted to allow successful execution of test cases.

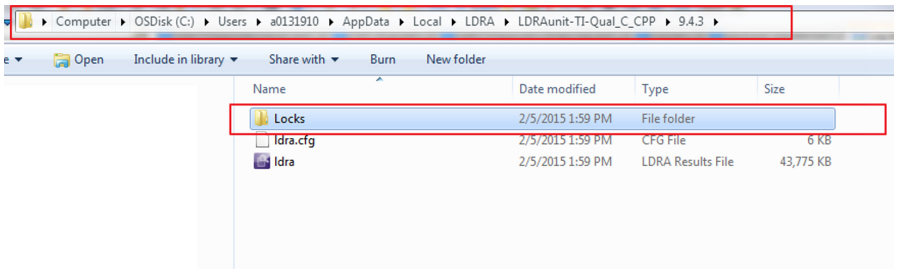


Figure 9. LDRA Analysis Lock

8 Steps for Using the Software Diagnostic Library TAU

Step 1: Open the Software Diagnostic Library TAU tool.

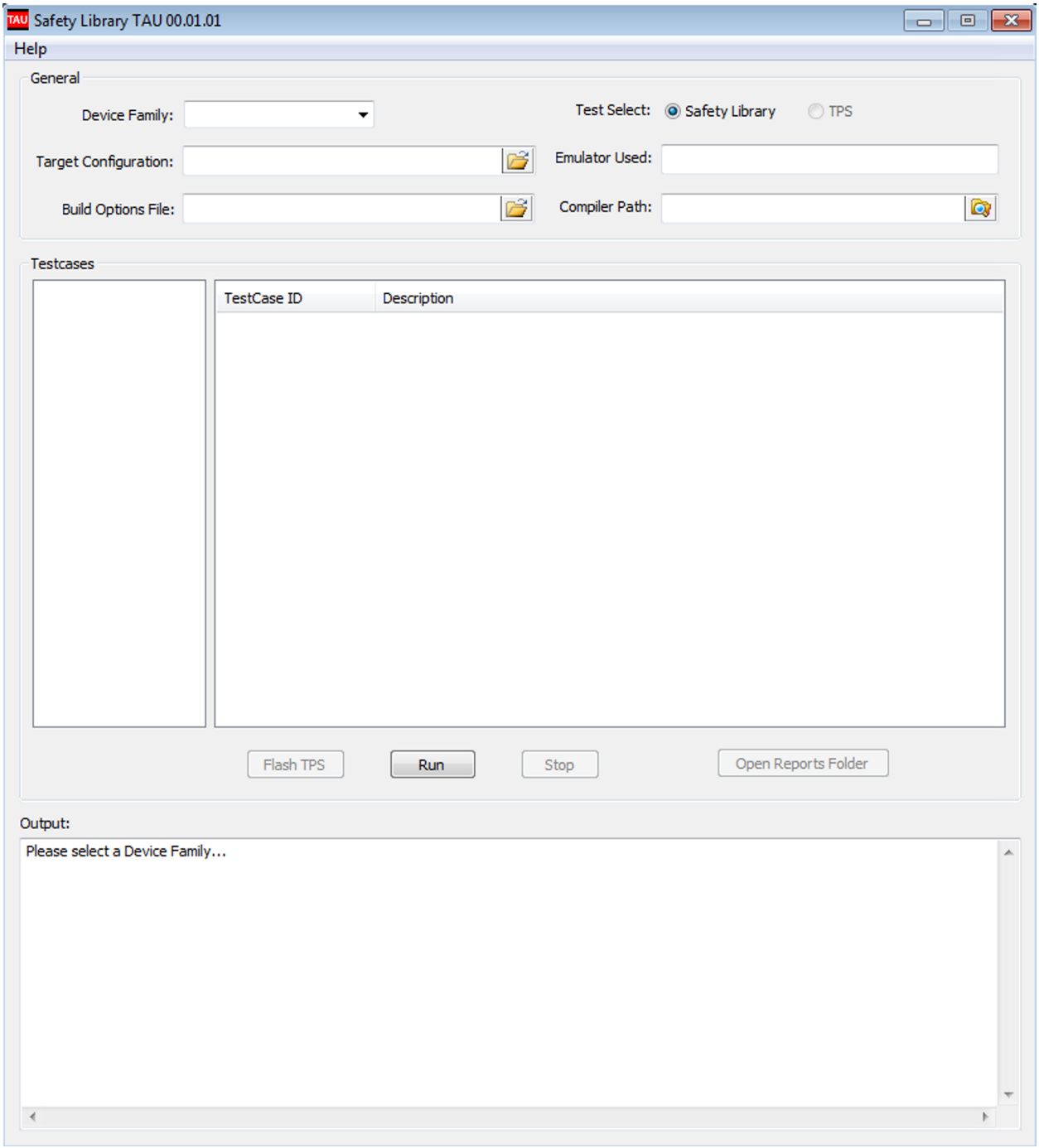


Figure 10. Software Diagnostic Library GUI Open Page

Step 2: Select a particular device from the Device Family drop-down menu.

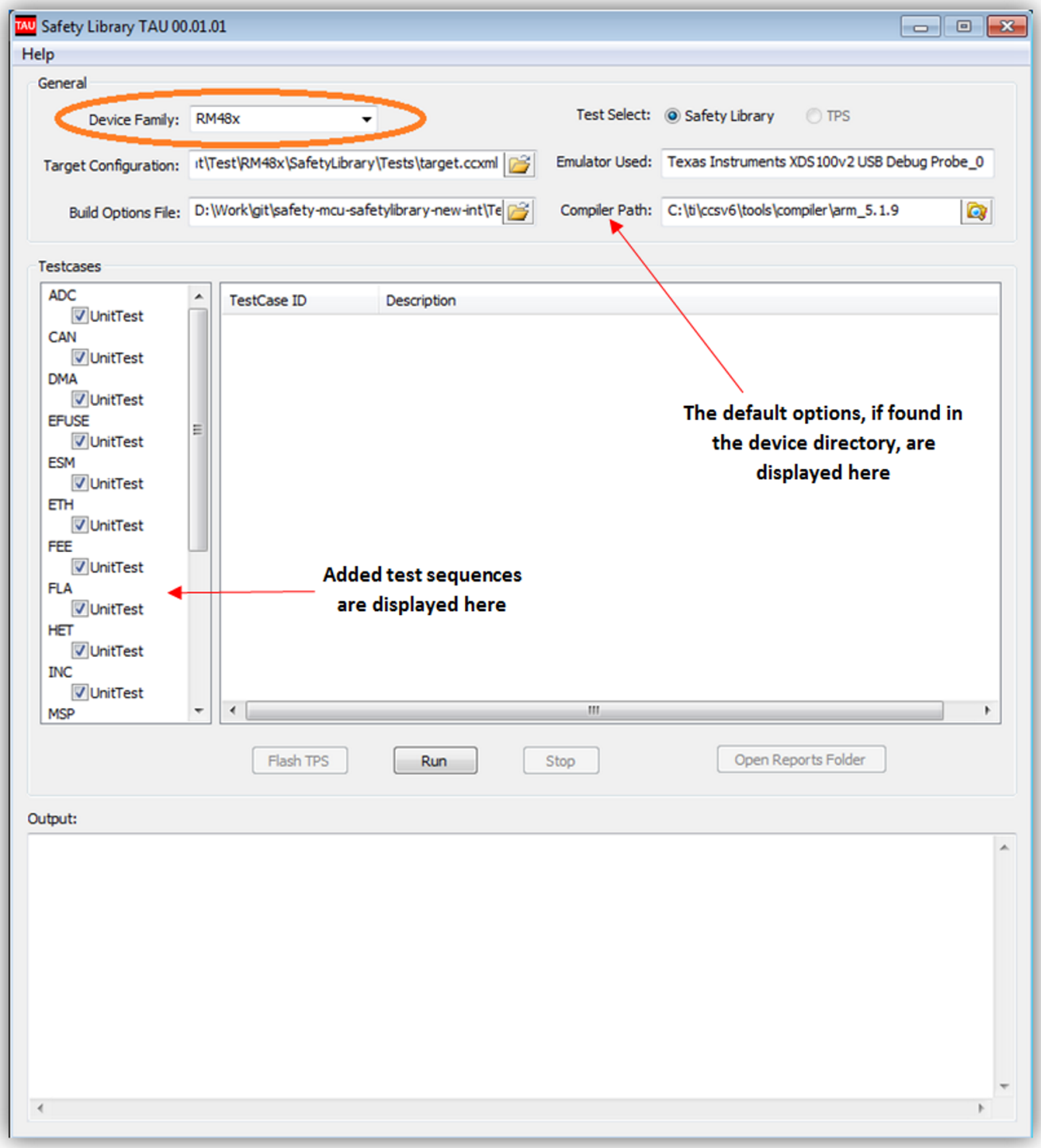


Figure 11. Device Family Selection

Step 3: Browse for the target configuration file and the build options file.

For more information about the target configuration file, see Section 9.2. For more information about the build options file, see Section 9.3.

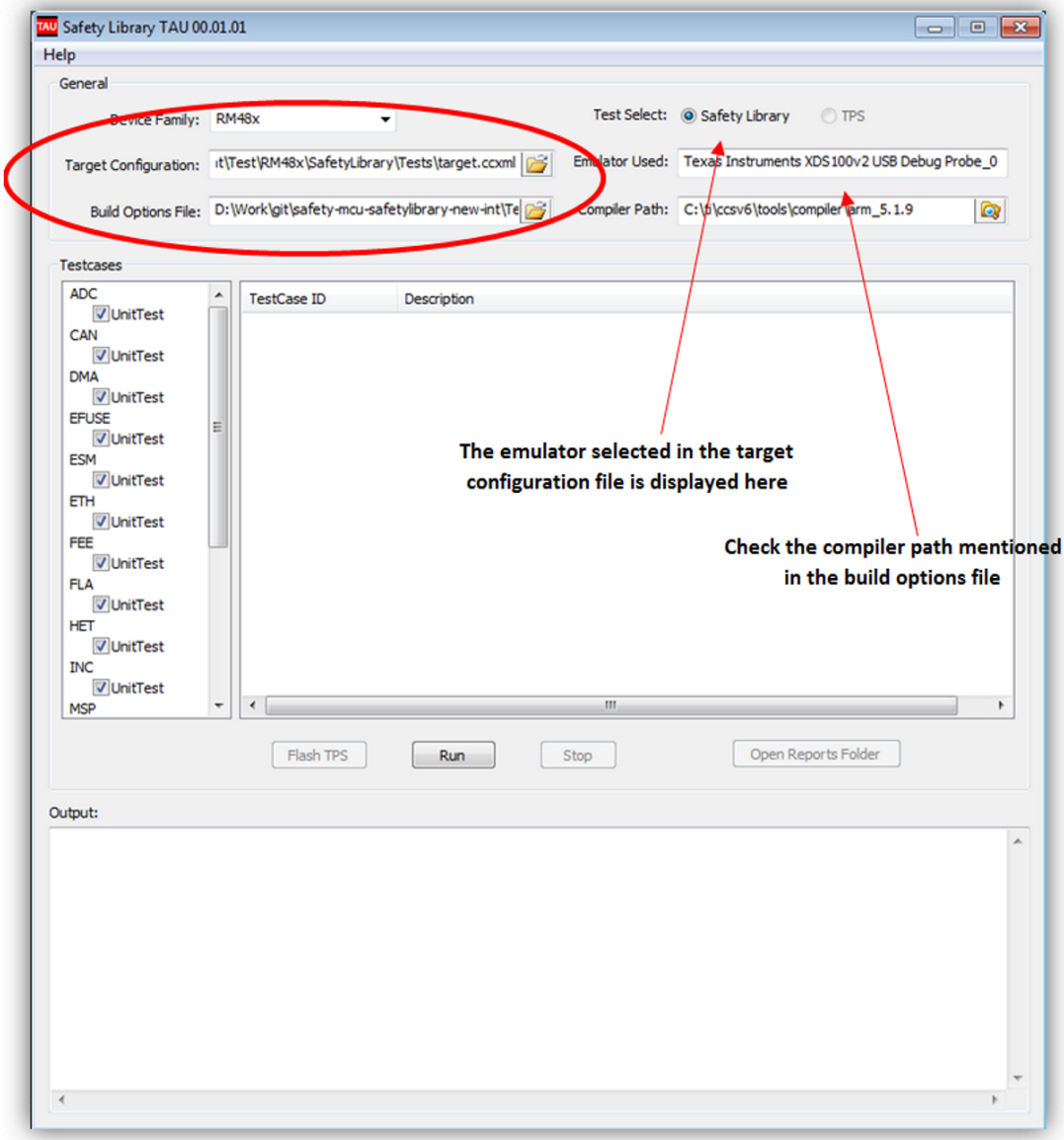


Figure 12. Target and Build Option Selection

Step 4: Select the tests to be run, then connect the board and click the “Run” button.

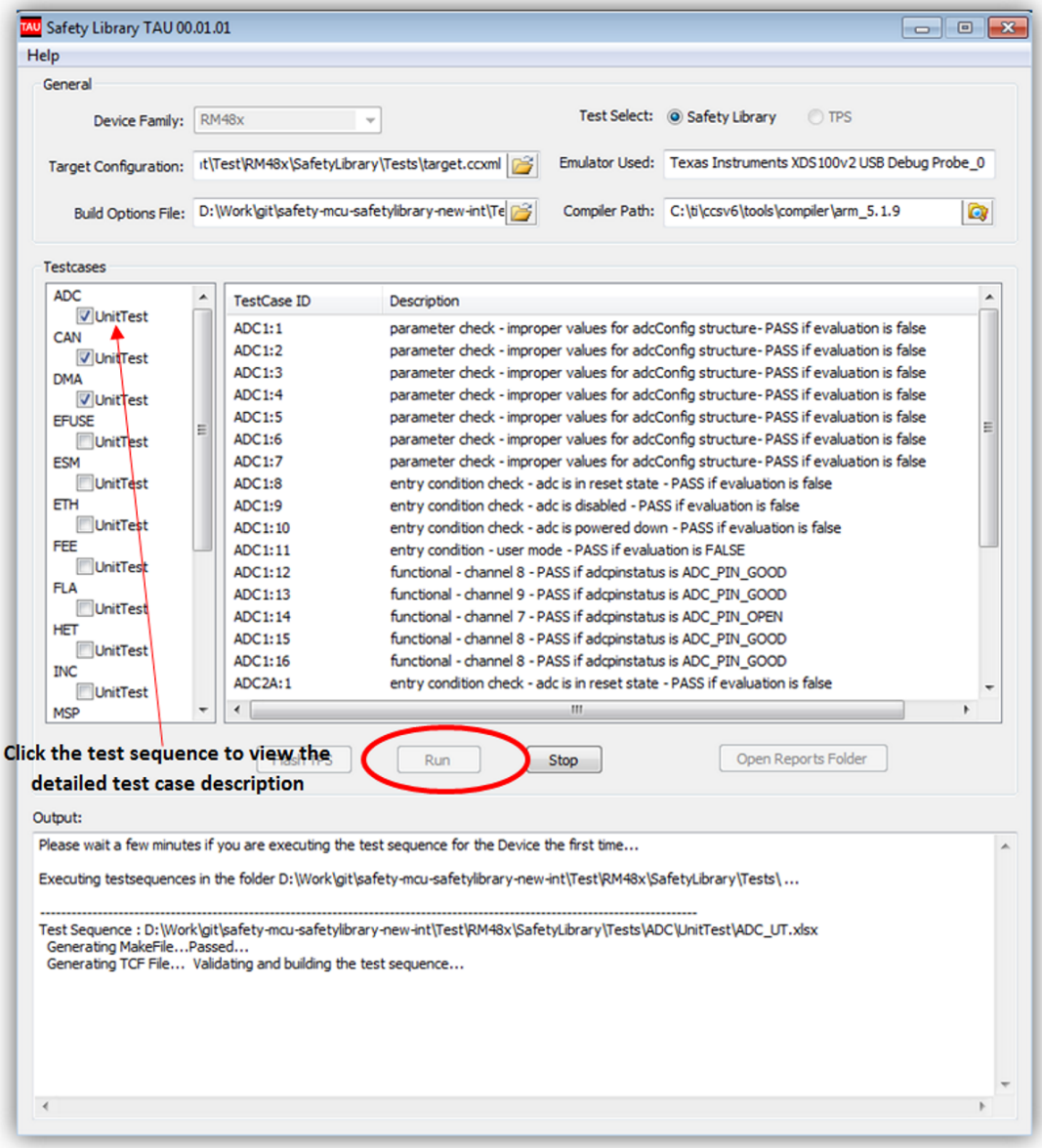


Figure 13. Test Case Selection and Run

When the build is successful and the .out file is created, the tool starts executing the test cases. The details of the test case execution is shown in a new pop-up window (see Figure 14).

Step 6: Test Execution

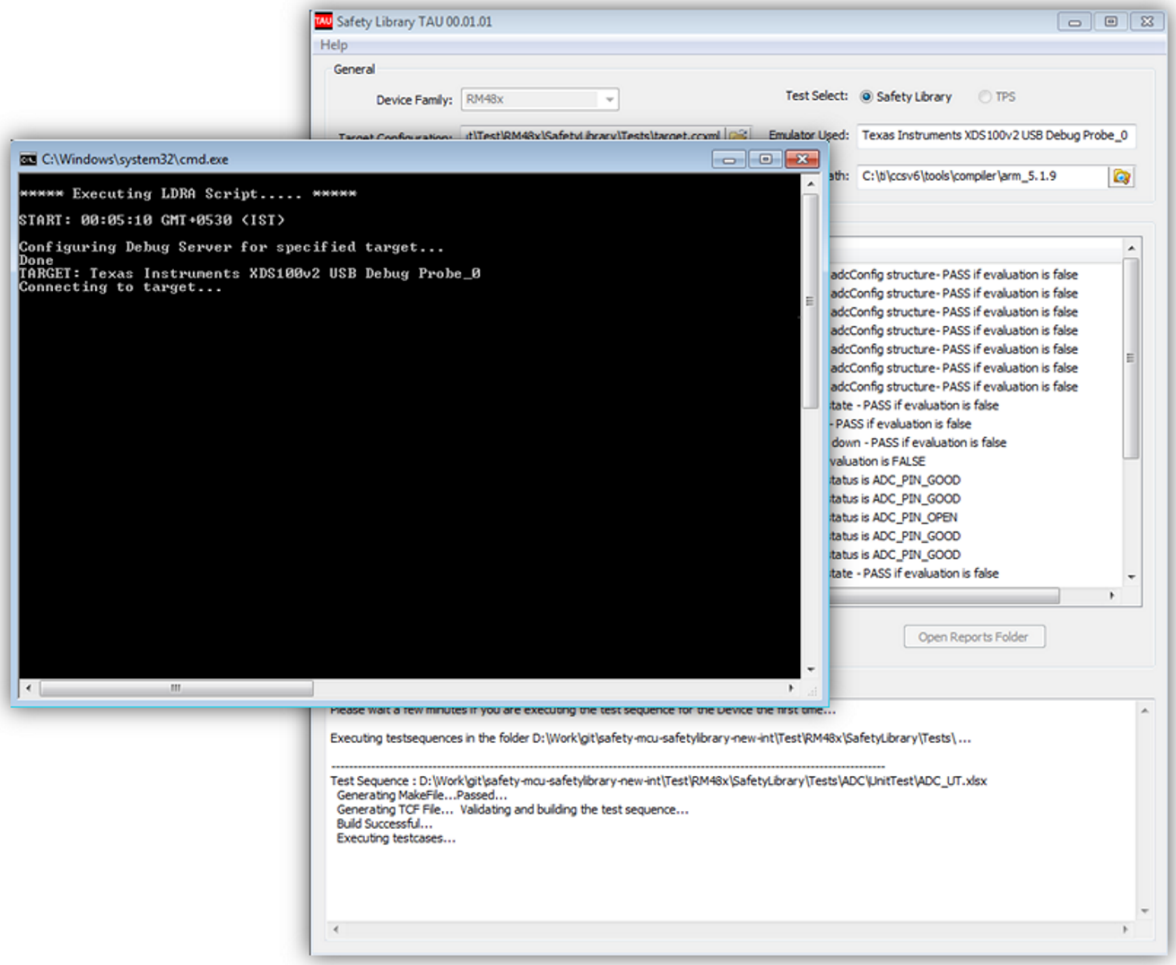


Figure 14. Test Execution

The generated reports are saved in the **<install_dir>\Test\<device>\SafetyLibrary\Reports** folder.

NOTE: Do not close the Software Diagnostic Library TAU window until the test execution is completed or successfully terminated after clicking the Stop button. Never kill the process while the test sequence is under analysis.

9 Inputs to Software Diagnostic Library TAU

9.1 Device Selection

Select the specific device to test the Software Diagnostic diagnostic library API on.

9.2 Target Configuration File

The target configuration file (.ccxml file) can be generated using the Code Composer Studio. Sample files are provided in the **<HALCoGen TAU install directory>\TargetConfiguration** folder. These sample files enable connection and flashing of the .out files to the device.

9.3 Build Options File

The build options file is a text file in the following format:

```
Compiler Options:
Linker Options:
Run time Library:
CG Tool Root Path:
COMPortNumber: 5
COMPortBaudRate: 9600
COMPortParity: N
COMPortDatabits: 8
COMPortStopBits: 2
```

NOTE: The options corresponding to the COMPort are default settings in the corresponding HALCoGen project (default SCI settings) of the device variant. If these settings are changed in the build options file, the test cases will not execute successfully unless the HALCoGen project configuration is also changed and the code is re-generated.

Some sample build options files are provided in the folder **<install directory>\Test\Misc\BuildOptions**. Users can use it as is in their project.

- **Compiler Options:**
  - ARM compiler options can be obtained from the appropriate device project file in the Code Composer Studio as shown in Figure 15 (Step 1).
- **Linker Options**
  - ARM linker options can be obtained from Code Composer Studio as shown in Figure 16 (Step 2).
- **Run Time Library**
  - Runtime support library is used for the CCS project as shown in Figure 17 (Step 3).
- **CG Tool Root Path**
  - The root path is where the CCS compiler is installed.
- COMPortNumber, COMPortBaudRate, COMPortParity, COMPortDatabits, COMPortStopBits
  - These options correspond to the SCI settings of the connected device.

NOTE: These settings must be in sync with the HALCoGen project corresponding to the device as in the demo_app\HALCoGen.

Step 1: Compiler Option Selection

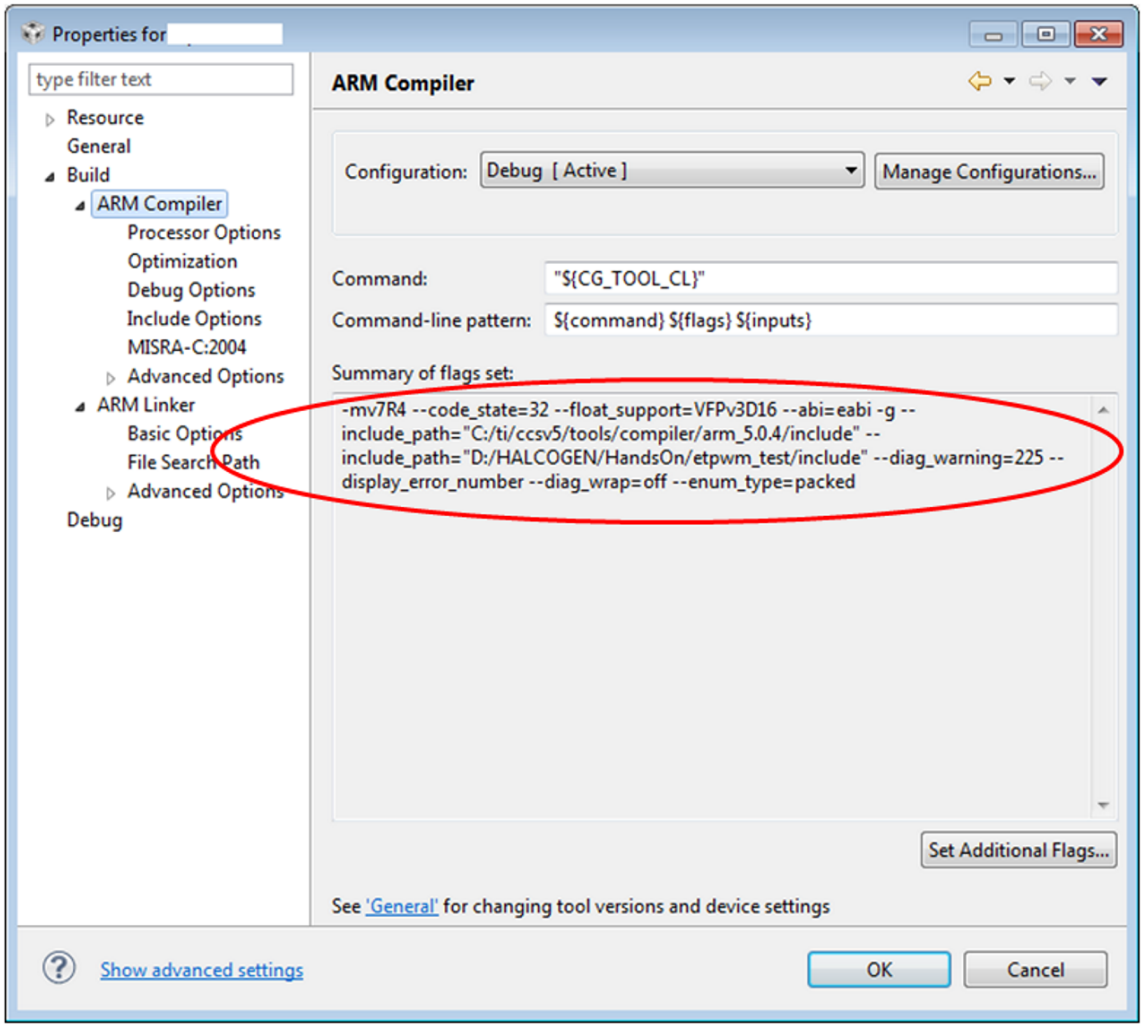


Figure 15. Compiler Option Select

Step 2: Linker Option Selection

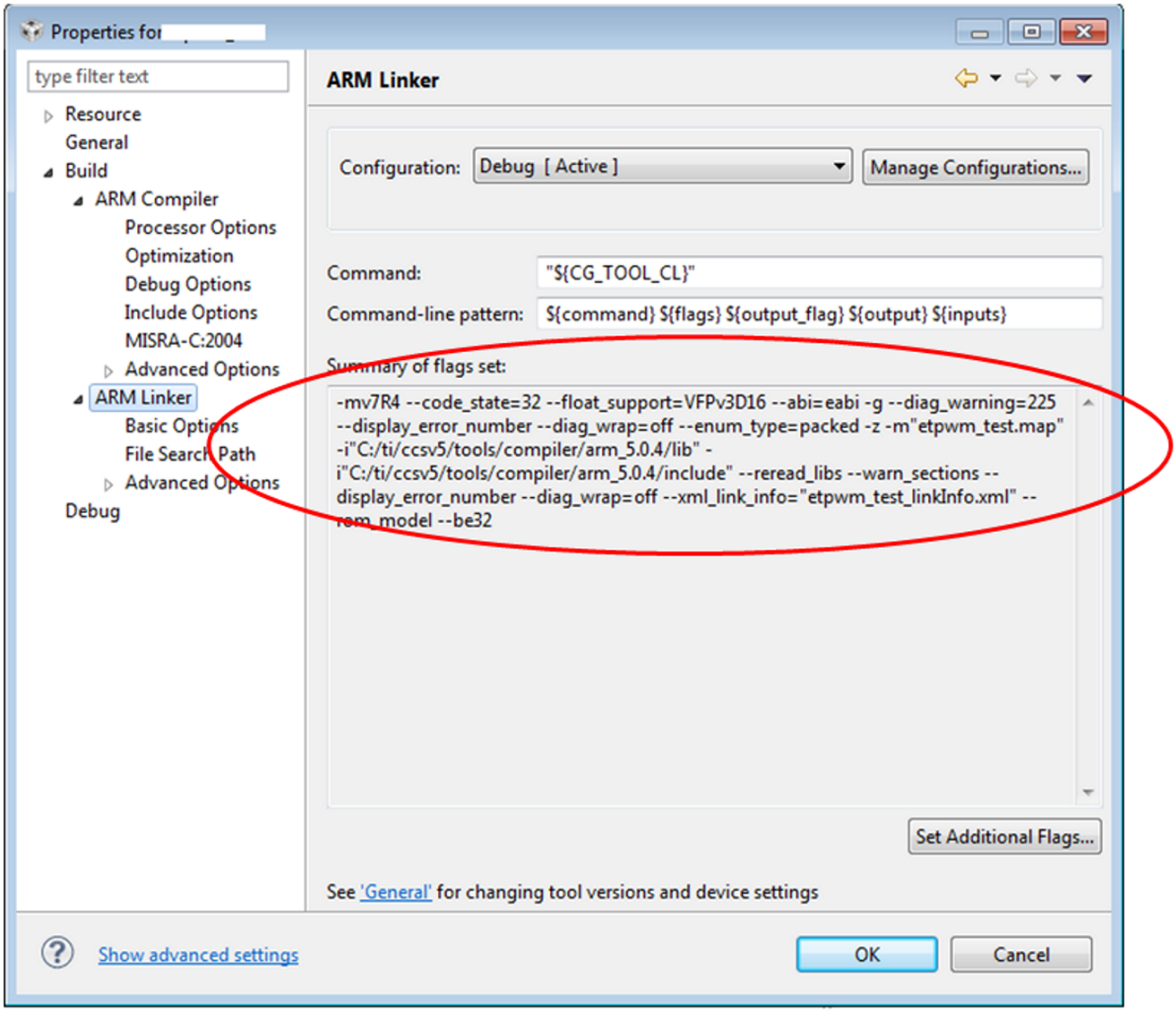


Figure 16. Linker Option Select

Step 3: Runtime Library Selection

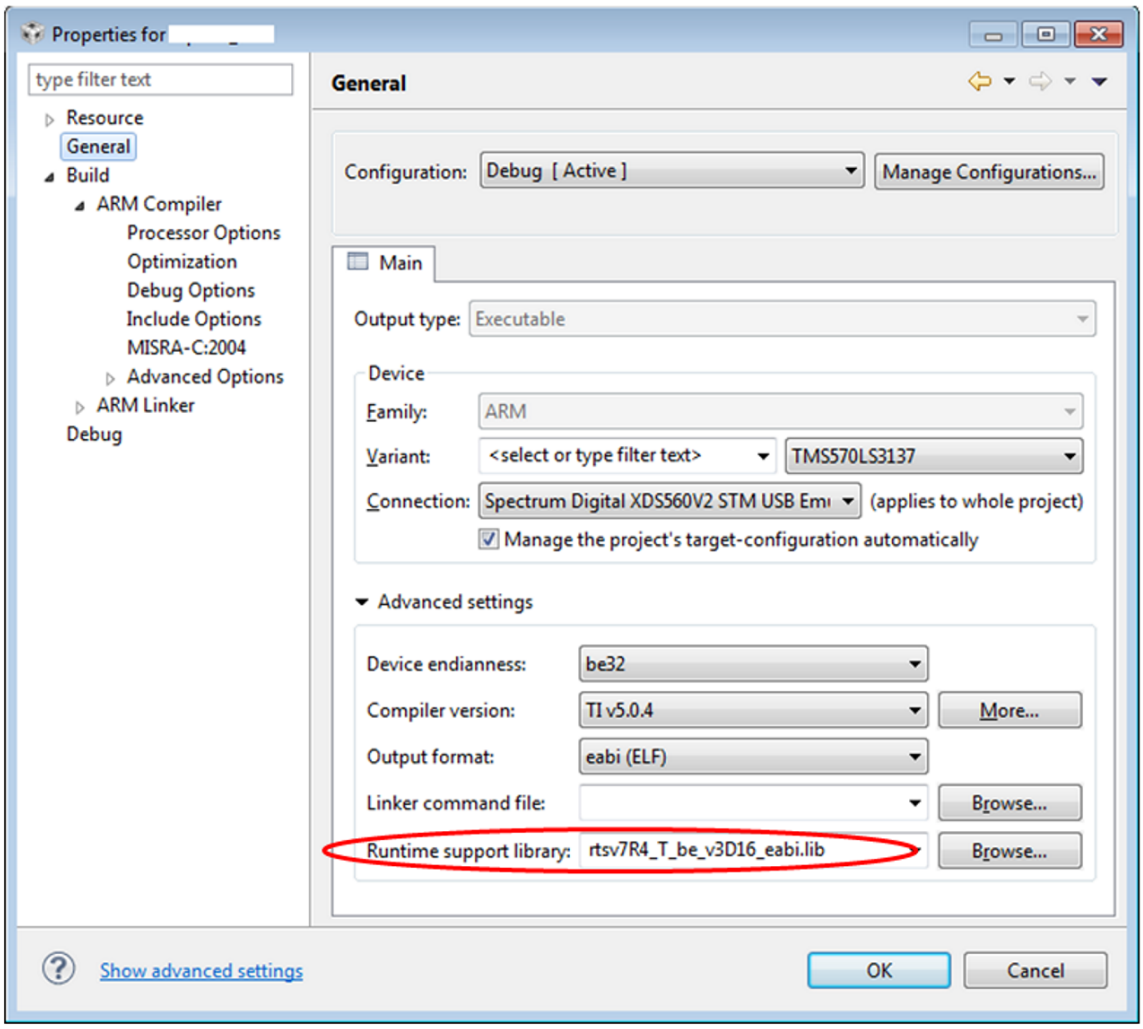


Figure 17. Runtime Library Selection

10 How to Add Individual Test Cases

Add a new folder and type “MyTests” in the test folder created by the tool. Define the test sequence and save it in the folder “Test1” inside MyTests. Figure 18 shows a template of a test sequence. Add more test sequences by adding more folders in MyTests.

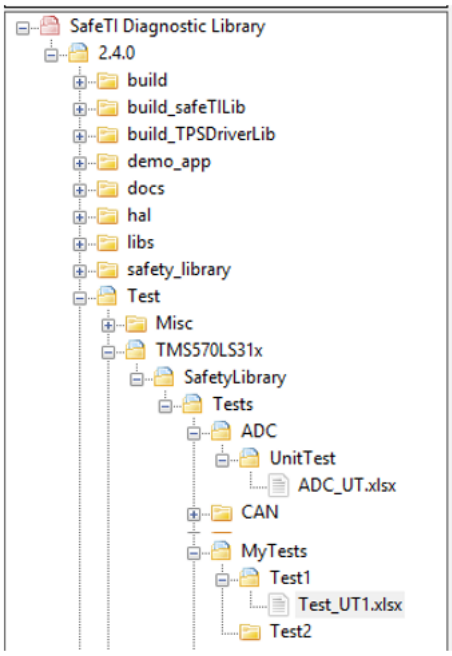
	1	2	3	4	5	6
1			TCF Description	<test sequence description>		
2			GlobalCode	<Global code for the test sequence> (Remove this row if not applicable)		
3			StartupCode	<Startup code for the test case> (Remove this row if not applicable)		
4			CleanupCode	<Cleanup code for the test case> (Remove this row if not applicable)		
5			TestCase Description	<description>		
6			TestCase ID & Requirement	<Test case ID> <Requirements>		
7	1	<filename>	Name	<parameter 1 name>	%	<global 1 name>
8		<function name>	Decl_type	<parameter 1 type>	<return type>	<global 1 type>
9			Df_type	Z	O	H
10		<no.of params + return + globals>	User_type	Input parameter applied through loc	Function result	Output global
11			Value	<parameter 1>	<expected value>	<expected value>
12						
13			UserDeclarations	<variable declarations if any> (Remove this row if not applicable)		
14			StartupCode	<Startup code for the test case> (Remove this row if not applicable)		
15			CleanupCode	<Cleanup code for the test case> (Remove this row if not applicable)		
16			TestCase Description	<description>		
17			TestCase ID & Requirement	<Test case ID> <Requirements>		
18	2	<filename>	Name	<parameter 1 name>	%	<global 1 name>
19		<function name>	Decl_type	<parameter 1 type>	<return type>	<global 1 type>
20			Df_type	Z	O	H
21		<no.of params + return + globals>	User_type	Input parameter applied through loc	Function result	Output global
22			Value	<parameter 1>	<expected value>	<expected value>
23						
24			End of Test Sequence			

Test case 1

Test case 2

Figure 18. Test Sequence Template

Folder structure as shown in Figure 19 must be followed strictly.



- New test case files/folders to be created as shown in the figure for ‘MyTests’ added for specific platform as per directory structure of installed CSP folder.
- Test case Excel file extension should be .xlsx, NOT .xls
- Each Folder should contain just 1 Test sequence (1 xlsx file)
- The sheet inside test.xlsx should be named ‘Test’. Other sheets in the workbook are ignored.

Figure 19. Folder Structure

NOTE:

- The Excel sheets must be saved as .xlsx and not .xls.
- One test folder must contain only one test sequence.
- The sheet containing the test cases must be named “Test”. Other sheets in the workbook are ignored.

Points to keep in mind while writing a test sequence:

- The test sequence must begin with “TCF Description” and end with “End of Test Sequence”.
- The global declarations and code must be inserted in the beginning of the test sequence after TCF description.
- User Declarations, Startup code, and Cleanup code (if needed) must be inserted above each test case.
- The test case description, ID, and requirements covered for each test case must be inserted above each test case.
- The total number of parameters of the test case (including the function input parameters, return value, and global variables checked) must be mentioned for each test case, as shown in Figure 20 and Figure 21.
- The file under test must be mentioned for each test case, and the function tested must be defined in each file.
- Two test cases must be separated by a blank row.
- End of the test sequence must be written in column 3 after the last test case. Leave a blank row after each test case.
- Each test sequence can test only one file.
- The Excel sheet must be saved as *_UT.xlsx (where UT stands for Unit Test). A code coverage report is also generated along with the regression report.

Figure 20 and Figure 21 show a few examples of a test case.

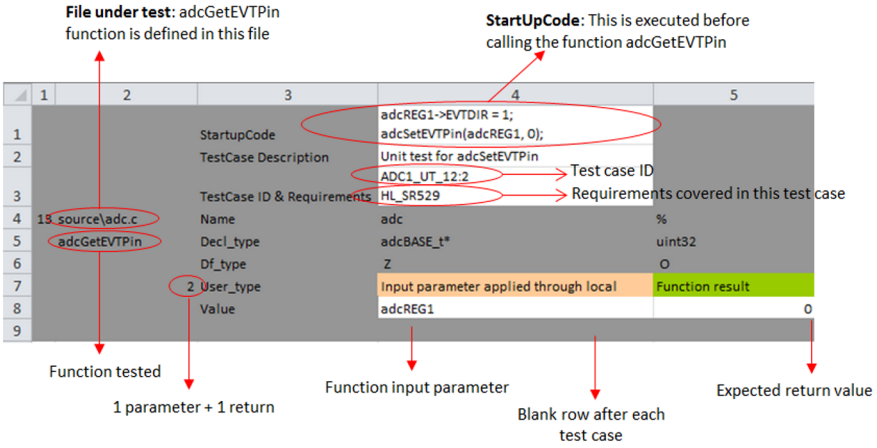


Figure 20. Test Case Example 1

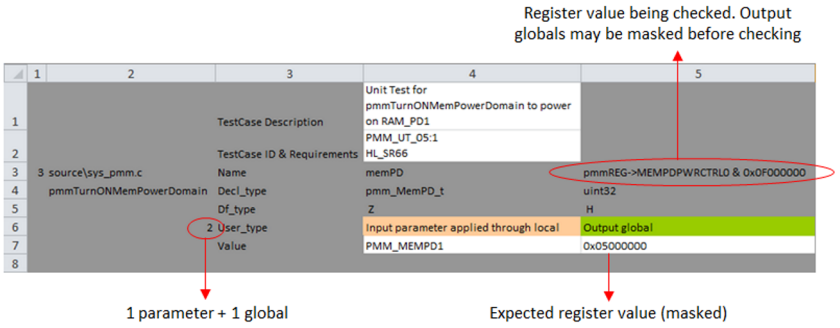


Figure 21. Test Case Example 2

Tip: Right-click on the test list field and click “Refresh” to regenerate the list.

11 Reports

The following sections define the reports generated at the end of test execution.

11.1 Regression Report

A regression report is generated for both unit and functional test sequences. The report is regenerated every time the test sequence is executed. No record of the previous test runs is stored.

11.2 Dynamic Coverage Analysis Report

A dynamic coverage analysis report is only generated for the unit test sequence. The following metrics are obtained in dynamic coverage analysis:

- Statement Coverage
- Branch and Decision Coverage
- MC and DC Coverage (Modified Condition and Decision Coverage)

Unlike the regression report, the dynamic coverage analysis report is an accumulated report of all the previous runs.

12 FAQ

1. The TCF generation fails when:
   - The test sequence is open in Microsoft Excel
   - A test case (including the last one) is not terminated by a blank row
   - The keyword “End of Test Sequence” is not found in the Excel sheet
2. What are the possible reasons for build failure?
   - License initialization failure
     - Check whether the LDRA license is properly installed or if it is expired
   - Validation failure
     - The file name mentioned is wrong or the function is not defined in the mentioned file name
     - The number of parameters for the test case mentioned in the Excel does not match the actual value

     NOTE: The number of parameters includes function input parameters, function return, and the global variables (or peripheral registers) checked.
   - Previous analysis was terminated in the middle of execution
     - Delete the analysis folder
   - Compile error
     - Users can check the CompileLog.txt in the **<test folder>\Debug** folder for the details
   - Invalid entries in build options file
3. Where do I see the instrumented code?
   - It is saved in LDRA work area.
4. The software says test execution is completed, but I cannot find the report in the reports folder. This happens when the tool was unable to connect to the target and execute the test.
   - Check whether the target is connected properly
   - Check the target configuration file

   In case of unit tests, the tool fails to generate the code coverage report if the test execution was terminated in the middle of execution.
5. Test execution stuck in the middle of execution. What to do? What is the reason?
   - To terminate the test execution, close the pop-up window. Never kill the process through the task manager. To stop executing the subsequent test sequences, click the “Stop” button and wait until the running process is completed. Do not close the TAU window before the process is completed.
   - The reason this happens is because of wrong configuration or wrong selection of functional test.
6. I use the full version of LDRA Tool Suite instead of LDRAUnit. Can I use this tool with this license?
   - Currently, the Software Diagnostic Library TAU supports only LDRAUnit.
7. How to Get LDRA Unit and Setting up the License After Downloading Software-Diagnostic-Library-CSP From TI?
   - After downloading Software-Diagnostic-Library-CSP LDRA Less package, the customer can contact LDRA and acquire the LDRAunit-TI-Qual 9.4.3 (or higher).
   - The customer can request for the LDRAUnit and full license from LDRA by sharing the following info with LDRA:
     1. Product Name: LDRAunit-SafeTI-CSP (C/C++, Windows)
     2. Exact part number: LPSN7W56SafeTI

- Instructions to procure LDRA:
  1. Contact LDRA at sales@ldra.com to request a quotation by sharing 'Product Name' and 'Exact Part Number' given above
  2. LDRA will issue a quotation and upon receipt of purchase order will send the customer a download link for the software and provide a license key.
  3. LDRA will set-up the web store page and provide a link to this so that customers can simply order and pay online.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265