HALCoGen-CSP Without LDRA

Contents

List of Figures

Trademarks

Hercules, Code Composer Studio are trademarks of Texas Instruments.
Microsoft Office is a trademark of Microsoft Corporation in the United States and/or other countries, or both.
Windows is a registered trademark of Microsoft Corporation in the United States and/or other countries, or both.
All other trademarks are the property of their respective owners.

1 Introduction

The HALCoGen compliance support package (CSP) was developed to provide documentation, reports and unit test capability to assist customers using HALCoGen generated software to comply with functional safety standards such as IEC 61508 and ISO 26262.

The HALCoGen **T**est **A**utomation **U**nit (HALCoGen TAU) is a tool that helps users generate a *Dynamic Coverage Analysis Report and Regression Report* for HALCoGen generated drivers to support ISO26262 and IEC61508 assessments.

HALCoGen TAU comes with the unit test cases and functional test case for all of the modules supported by HALCoGen for the Hercules™ family of devices and the necessary test infrastructure to run these test cases. HALCoGen TAU also provides infrastructure for Hercules customers to add their own test cases for specific used cases.

2 Software Requirements

- Windows® Windows Operating System version 7 or higher
- HALCoGen version 04.02.00 or higher (highly recommend see FAQ).Section 12.7
- Perl 5.x. Download Link → http://www.perl.org/get.html#win32
- Code Composer Studio™ 5.4 (or higher)
- Microsoft Office™ 2007 (or higher) for Microsoft Excel used during LDRA Tests
- 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 (See Section 12.15)

3 HALCoGen TAU Tool Restrictions

- This tool supports HALCoGen projects generated with TI tools only.
- This tool supports device families TMS570LS31x/21x, RM48x, TMS570LS12x/11x, RM46x, TMS570LS04x/03x, RM42x, TMSLS09x/07x, RM44x, TMS570LC43x, and RM57x only.
- This tool does not support testing FEE module on device families TMS570LS04x/03x and RM42x.
- This tool does not support testing USB module.
- This tool does not support testing of assembly files.
- This tool does not support testing of FREERTOS driver.

4 HALCoGen CSP Contents

4.1 Below is a List of Items Supplied With the HALCoGen Compliance Support Package

- Requirement Document
- Software Architecture Document
- Design Document
- Software Safety Manual
- Dynamic Coverage Analysis Report
  - Statement Coverage
  - Branch Coverage
  - MC/DC Coverage
- Test Manager report
  - Quality Review Report - HIS Metrics
  - Code Review Report - MISRA C
  - Regression Report
- Test Results Reports
  - Unit Test Report
  - Safety Functional Tests
- Traceability Report
- Software User's Guide
- Software Release Note
- HALCoGenTAU
  - Graphical user interface (GUI) to choose and run the test in the user’s environment on target hardware
  - Unit level and functional test cases

Since HALCoGen is a GUI configuration-based code generator tool, providing a static report to customers may not be suitable for the assessment since the source code is dynamic.

The HALCoGen TAU provides the options for the user to generate the Dynamic Coverage Analysis Report and Regression reports at their premises for their specific configuration. The resulting customized reports can then be used to assist customers in their end system safety assessments.

By default, the documents and test reports can be found in C:\ti\Hercules\SafeTI-HALCoGen-CSP\<Device_Family_version>\HALCoGen\<Device_family>\CSP.

5 Terminologies Used in HALCoGen TAU

5.1 What is Unit Testing?

In simple words, it is single function tested in isolation. Unit testing generally involves taking a specific subset of the software, linking it with a test driver and exercising it, checking that it behaves as expected under all conditions. This subset of the software is generally a single function. The source file under test is instrumented and tested in white box mode to get the code coverage. This is white box mode test because to develop unit test cases for the function under test, the function's usage, internal working and parameters used are all required to be understood.

5.2 What is Functional Testing?

Functional testing involves testing more than one function tested together to verify the if the desired set of operations like IO sequences is properly executed. The source file(s) under test will not be instrumented as the target of test is not specific function or subset in file and is tested in black box mode. This is black box mode test because to develop Functional test cases, knowing the functionality to be tested and functions' usage alone are sufficient without any knowledge of function internal operations.

5.3 What is a Test Sequence?

A test sequence is a set of test cases, unit or functional (not both) targeted on a single or set of c files. A test sequence is written in the form of 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, to 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 a TCF file, which is the actual input to LDRA unit. Figure 1 shows an example of a test case in the test sequence. Click here to see how to add your own test cases.

	1	2	3	4	5
1			StartupCode	adcREG1->EVTDIR = 1; adcSetEVTPin(adcREG1, 0);	
2			TestCase Description	Unit test for adcSetEVTPin	
3			TestCase ID & Requirements	ADC1_UT_12:2 HL_SR529	
4	13	source\adc.c	Name	adc	%
5		adcGetEVTPin	Decl_type	adcBASE_t*	uint32
6			Df_type	Z	O
7			2 User_type	Input parameter applied through local	Function result
8			Value	adcREG1	0
9					

Figure 1. Excel Test Case Snapshot

Each test case executes in the following sequence:

- 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

5.4 What is a Test Case File (TCF)?

A Test case file (TCF) contains all of the information required to run and re-run the test cases. The sequences are converted into a TCF. TCF contain the tags for the test case ID's and requirements ID's, which help provide traceability. LDRA unit can group TCF's with regression reports and they can be stored for regression verification by either saving it with the source file, via a software configuration management (SCM) system, or 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. TCF's can also be re-run from the command line and in batch mode so that as the source code changes module interfaces and output can be verified.

5.5 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 HALCoGen TAU uses LDRAunit in the background to generate the following code coverage criteria:

- Statement Coverage
- Branch coverage
- MC/DC Coverage

5.6 What is a Regression Report?

A Regression report is the consolidated test results report generated by LDRA unit running the functional and unit tests selected through HALCoGen TAU.

6 Functional Blocks of HALCoGen TAU

The functional blocks of HALCoGen TAU are shown below:

- LDRAunit-TI-Qual
  - Helps in generating the dynamic analysis report
  - Interfaces to CCS debug server scripts
- CCS Debug Scripts
  - Helps in loading and executing the test cases
- TI Test Cases
  - Excel-based unit and functional test cases per module driver supported in HALCoGen
- TI Test Script Engine
  - Generates HALCoGen driver files from the selected HALCoGen project
  - Instruments the targeted C file through LDRA
  - Generates TCF files, invoking LDRAunit
  - Generates the executable through auto generated make file
  - Helps in consolidating the code coverage report and regression report generated by LDRA
- HALCoGen TAU GUI
  - GUI to help choose:
    - HALCoGen project
    - HALCoGen version
    - Build options based on device
    - Target configuration based on boards and debugger
    - Add test cases from the test repository
    - Option to choose the test cases or test sequences
    - Update the status information of each and every test sequences selected

7 HALCoGen TAU Test Flow

Figure 2 and Figure 3 show the typical automated test flow followed by the HALCoGen TAU.

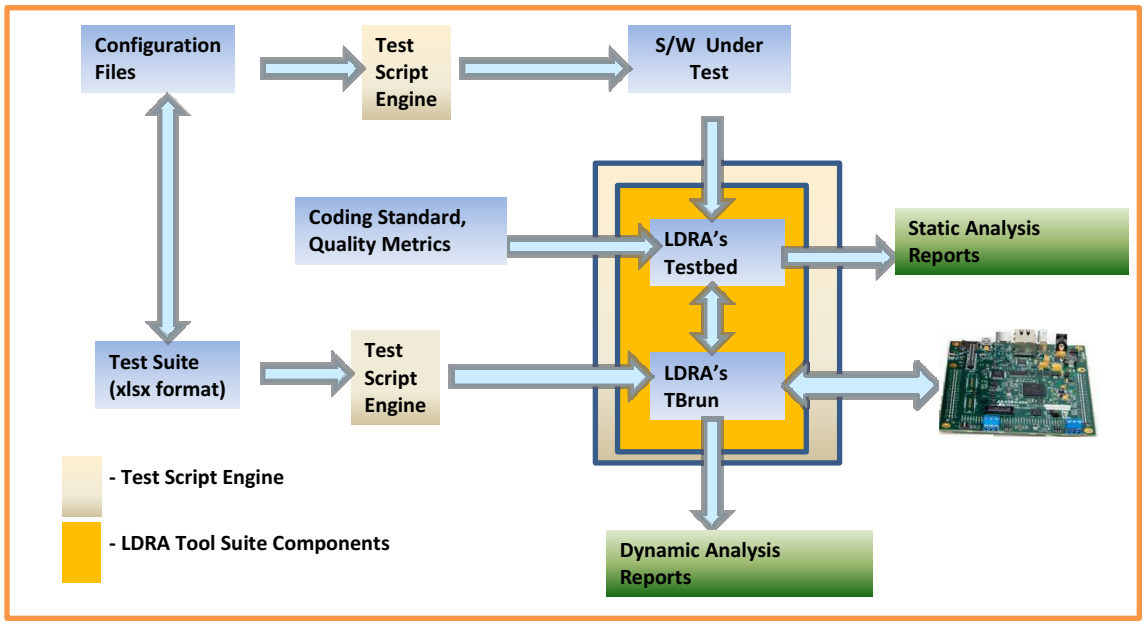

Figure 2. Automated Static and Dynamic Analysis Flow

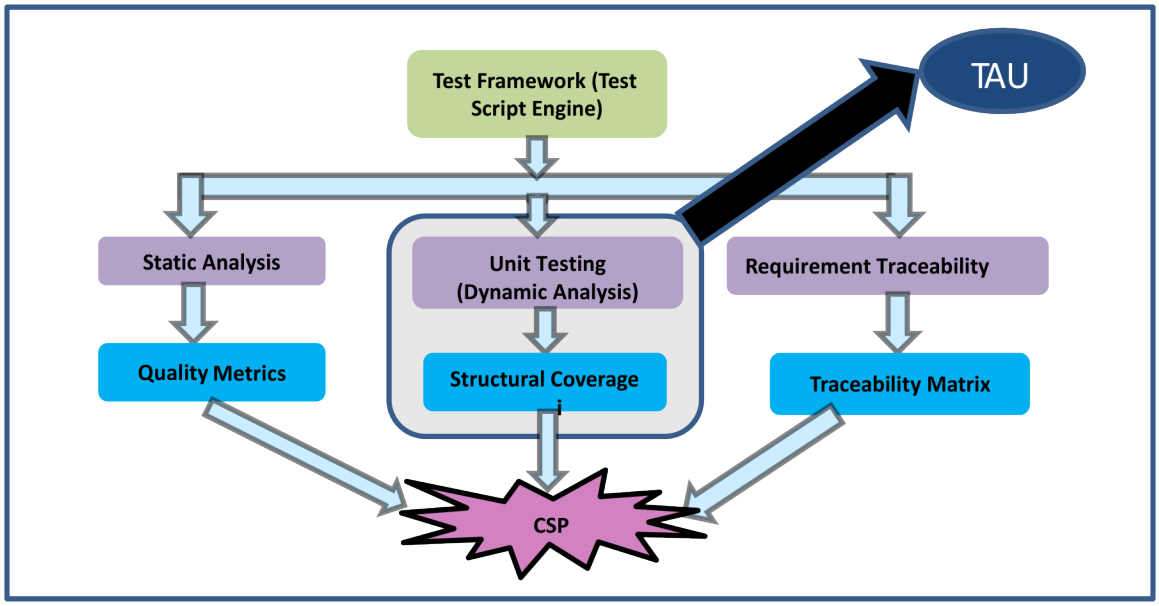

Figure 3. Test Automation Framework for CSP

8 How to Use HALCoGen TAU? A Walkthrough

1. Create a HALCoGen project using HALCoGen (v04.02.00 or higher) (see Figure 4).

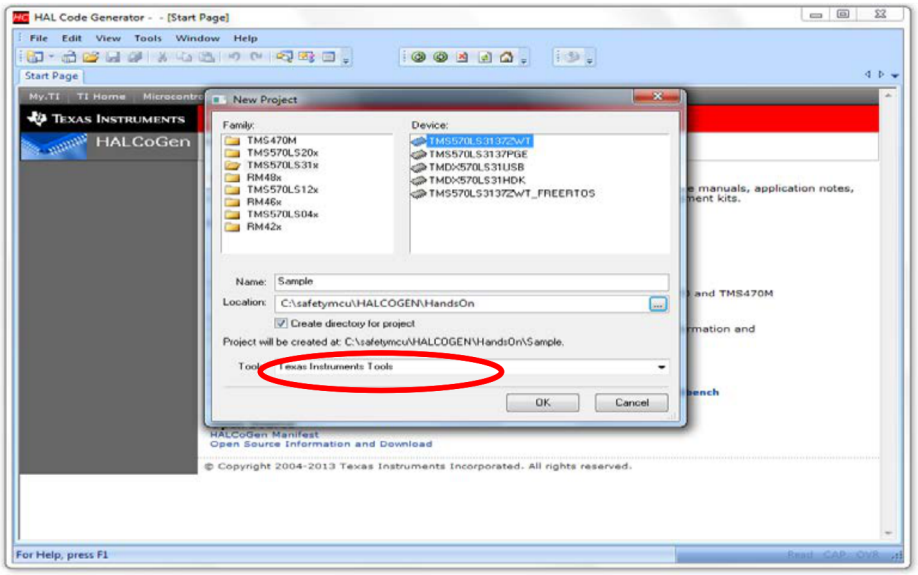

Figure 4. HALCoGen Project Selection

2. Do the necessary configuration, save the project and close HALCoGen (see Figure 5).

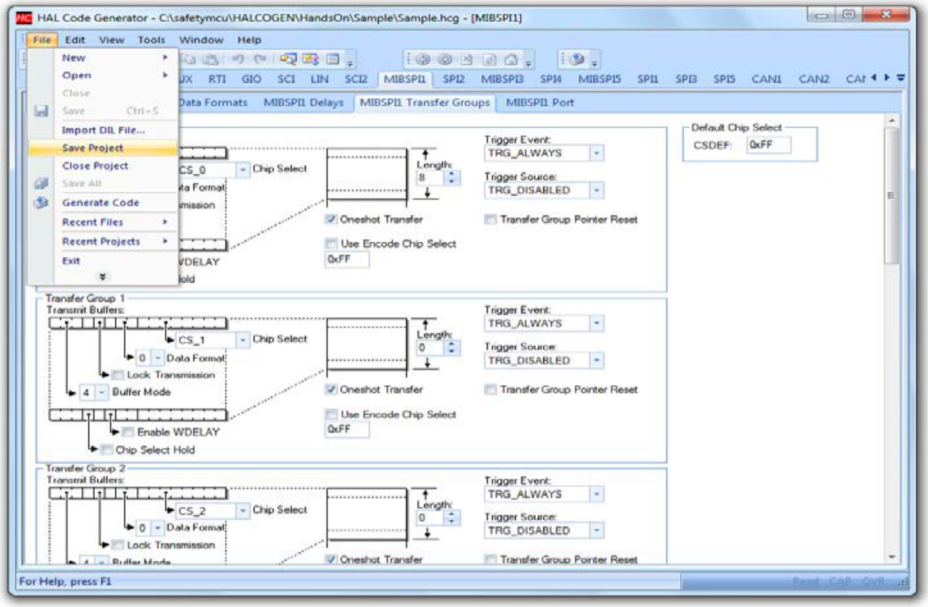

Figure 5. HALCoGen Module Configuration

3. Open the HALCoGen TAU tool (see Figure 6).

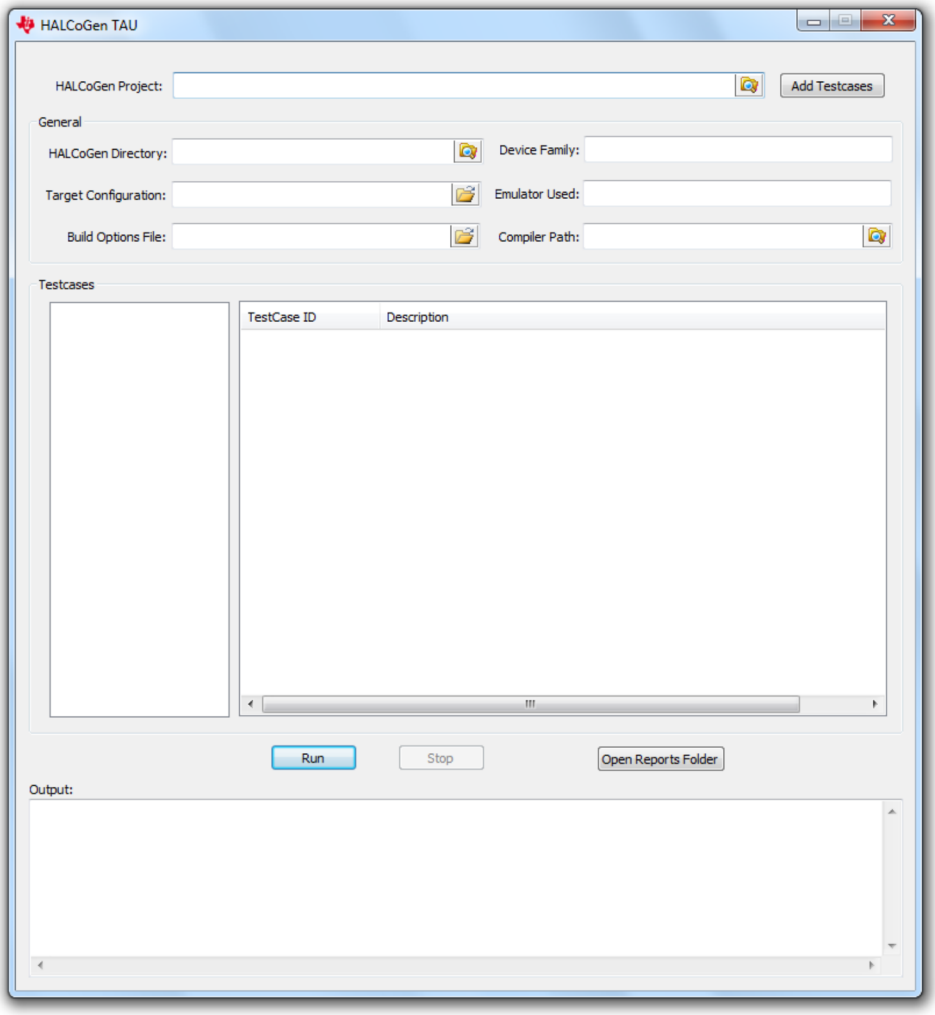

Figure 6. HALCoGenTAU GUI Open Page

4. Browse the folder containing the HALCoGen project and click the *Add Testcases* button (see Figure 7).

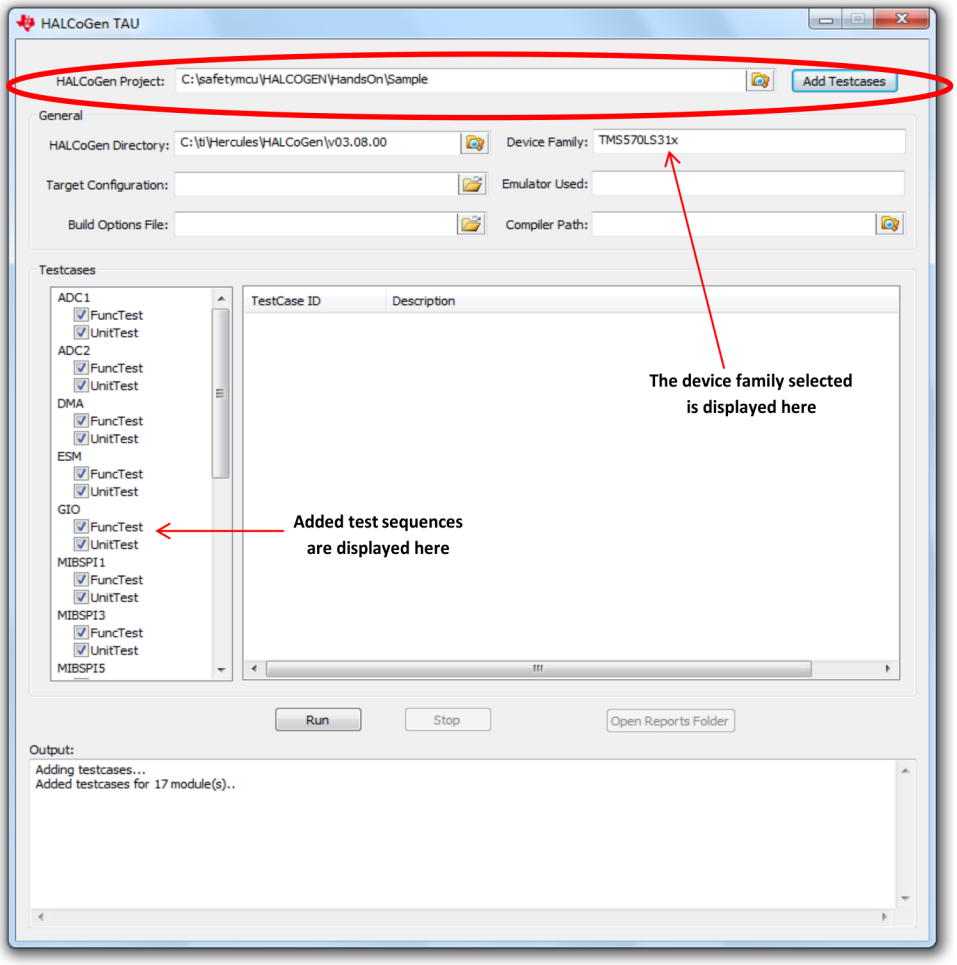

Figure 7. HALCoGen Project Selection and Add Test Case

Clicking the *Add Testcases* button adds a new folder *Test* in the selected HALCoGen Project folder. The test cases for the enabled modules are added to the folder.

5. Browse for the HALCoGen directory, target configuration file and build options file (see Figure 8).
   Click here to know more about target configuration file and build options file. (Sample target configurations files and the build options file are provided in the *<install_dir> \TargetConfiguration* and *<install_dir>\BuildOptions* folders, respectively).

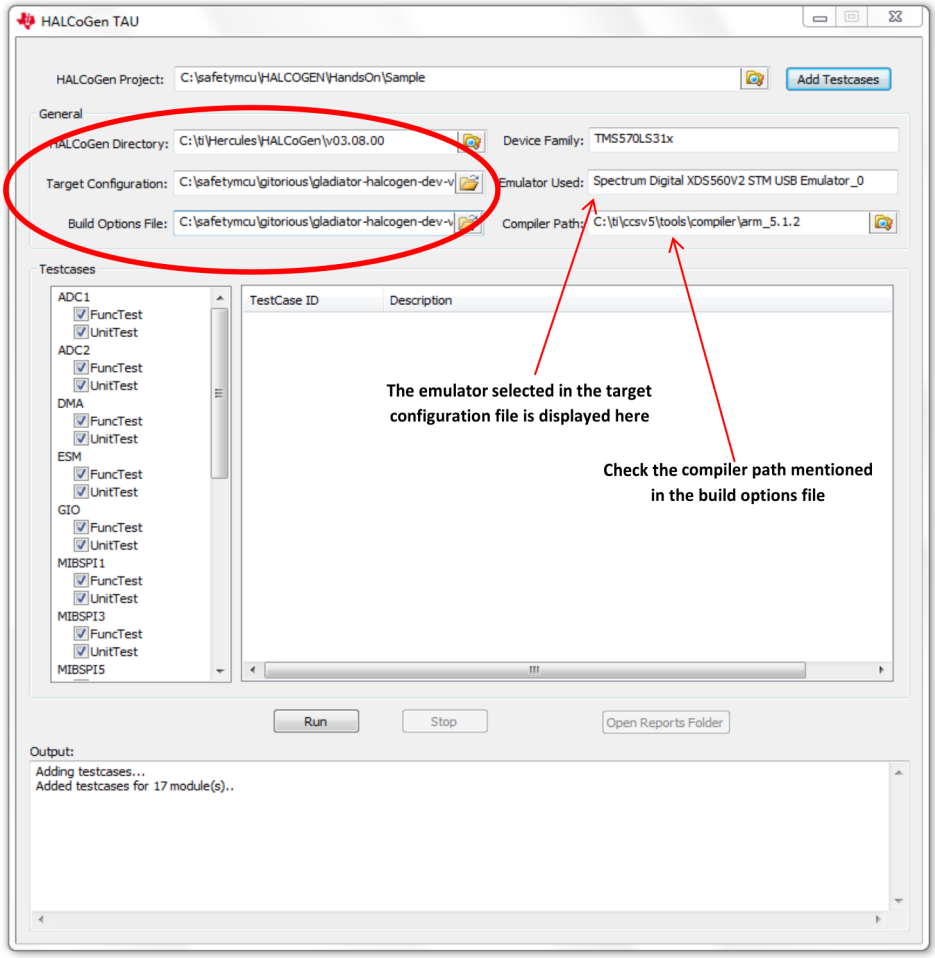

Figure 8. Target and Build Option Selection

6. Select the tests to be run and click the *Run* button (see Figure 9).

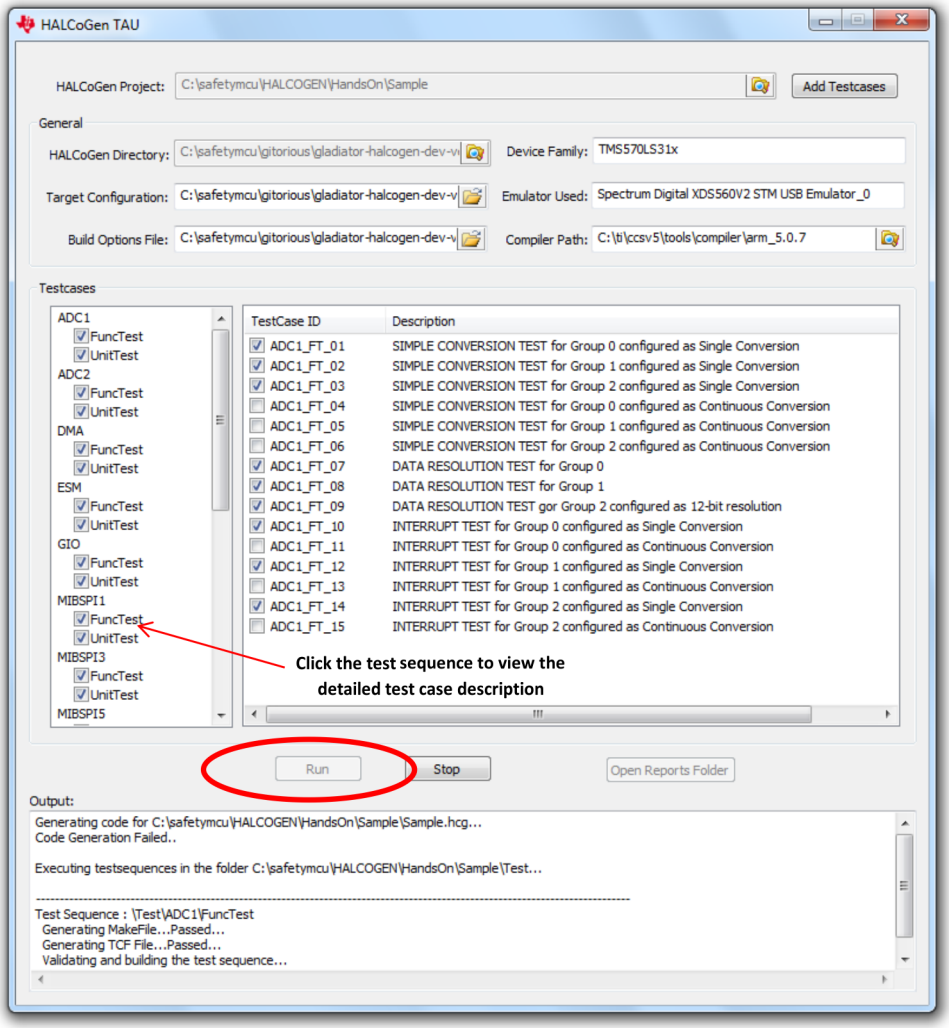

Figure 9. Test Case Selection and Run

Once the build is successful and .out is created, the tool starts executing the test cases. The details of the test case execution is shown in a new pop-up window

NOTE: Make sure the target Board is powered and connected with JTAG Emulator before hitting Run.

7. Test Execution (see Figure 10).

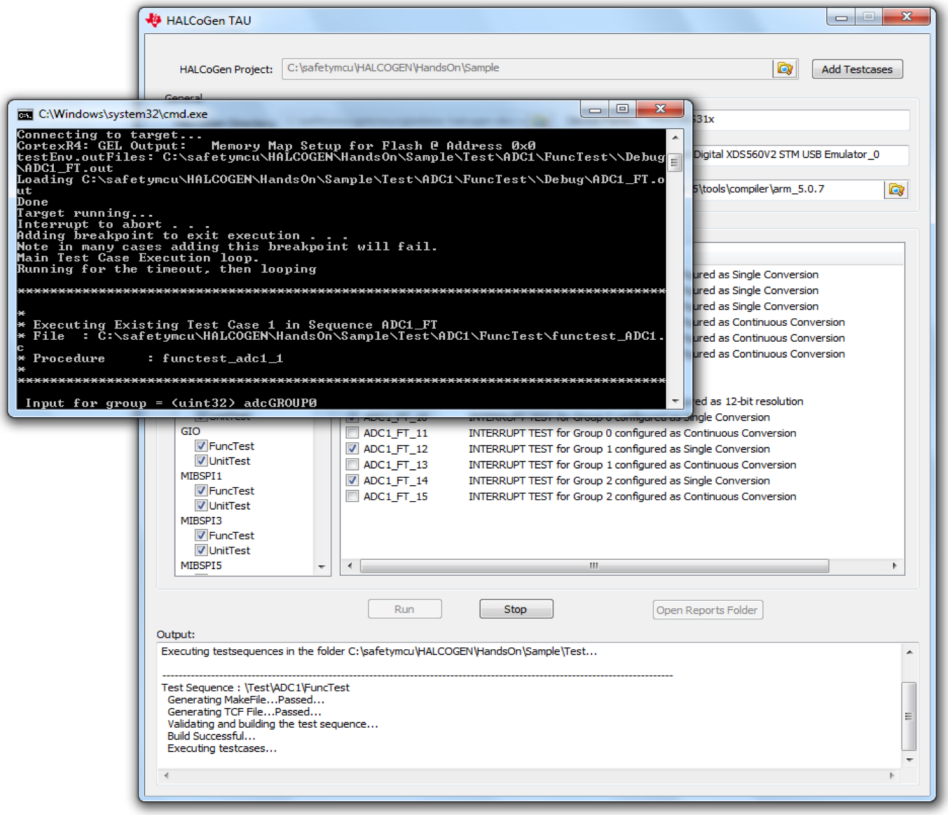

Figure 10. Test Execution

The reports generated are saved in the <*HALCoGen project folder*>*Reports* folder.

NOTE: Do not close the HALCoGen 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.

A copy of the build options file and the target configuration file is saved in the “Test” folder inside your HALCoGen project folder. The next time you open this project, these fields will be automatically updated with these files.

HALCoGen TAU will compile files within the <*HALCoGen_Project*>*source* folder generated by HALCoGen and the respective test folder. The test folder can be UnitTest, FuncTest or the custom folder added by the user inside the <*HALCoGen_Project*>*test*\<*Module*> folder.

9 Inputs to HALCoGen TAU

9.1 HALCoGen Project Folder

The folder containing the HALCoGen project.

NOTE: Folder name should not contain any spaces.

9.2 HALCoGen Directory

HALCoGen installation path.

9.3 Target Configuration File

The target configuration file (.ccxml file) can be generated using Code Composer Studio.

Sample files are provided in the <*HALCoGen TAU install directory*>*TargetConfiguration* folder.

9.4 Build Options File

Build Options file is a text file in the following format:

```
Compiler Options:
Linker Options:
Run time Library:
CG Tool Root Path:
```

Compiler Options: Linker Options: Run time Library: CG Tool Root Path: <*HALCoGen TAU install directory*>*BuildOptions*. Users can use it as is in their project.

- Compiler Options:

  ARM compiler options can be obtained from Code Composer Studio‘s appropriate device project file as in Step1.
- Linker Options

  ARM linker options can also be obtained from Code Composer Studio as in Step 2.
- Run time Library

  Runtime support library used for the CCS project as in Step 3.
- CG Tool Root Path

  The path where CCS compiler is installed

1. Select compiler option (see Figure 11).

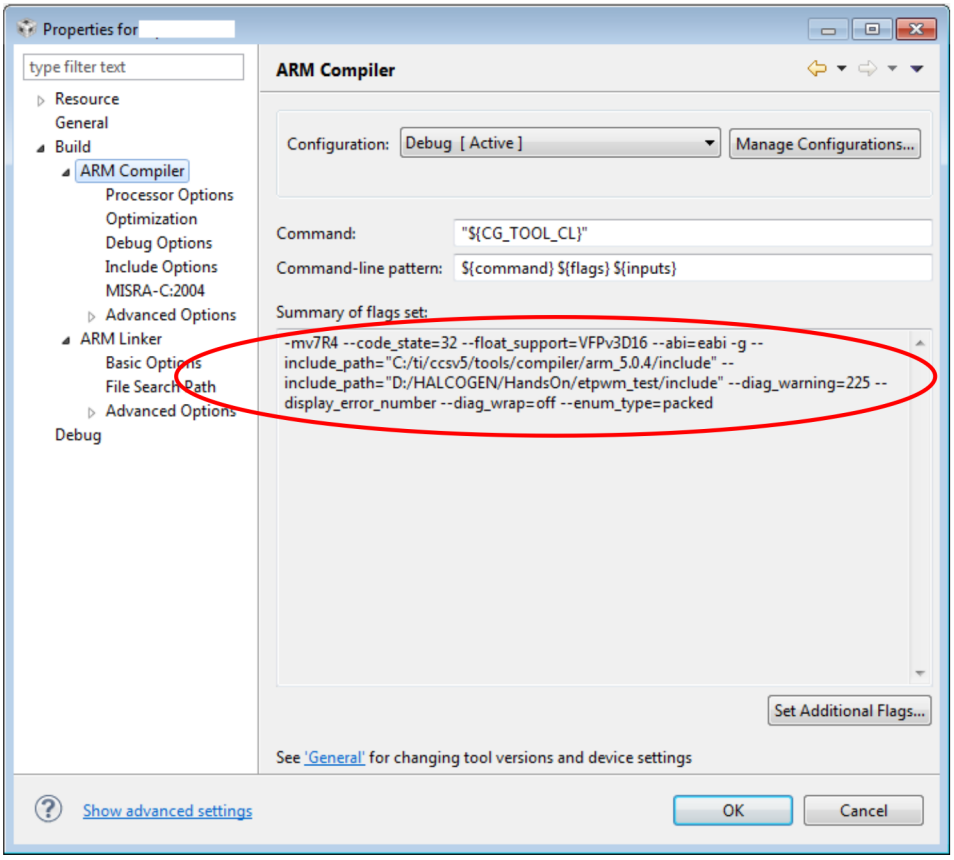

Figure 11. Compiler Option Select

2. Select linker option (see Figure 12).

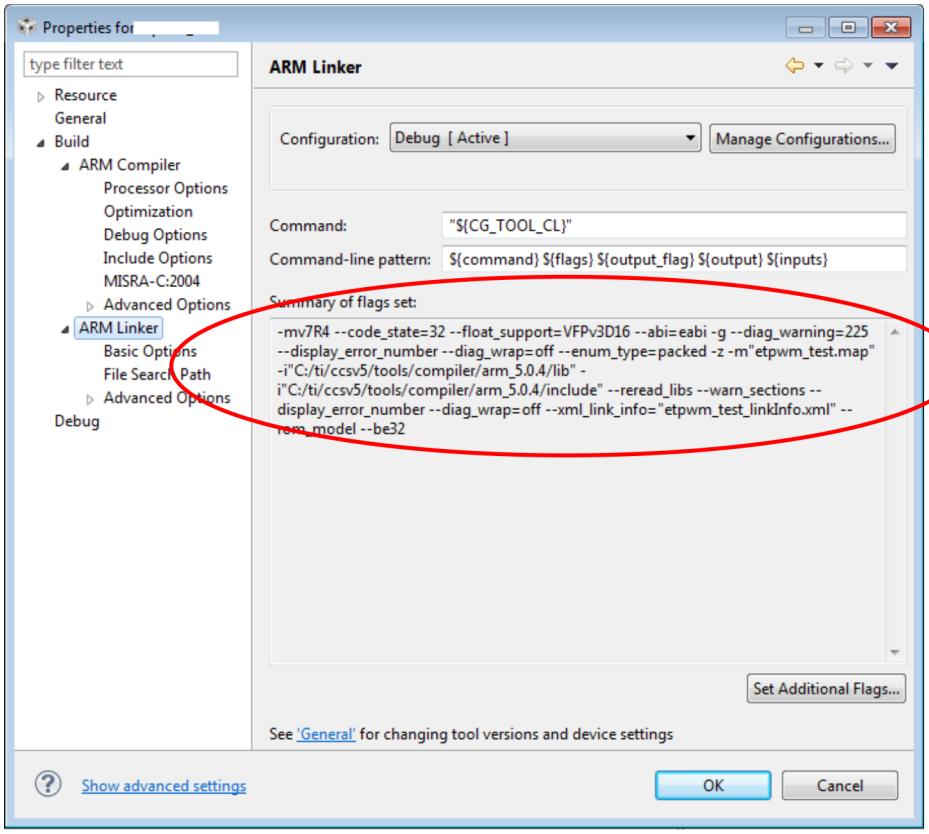

Figure 12. Linker Option Select

3. Select runtime Library (see Figure 13).

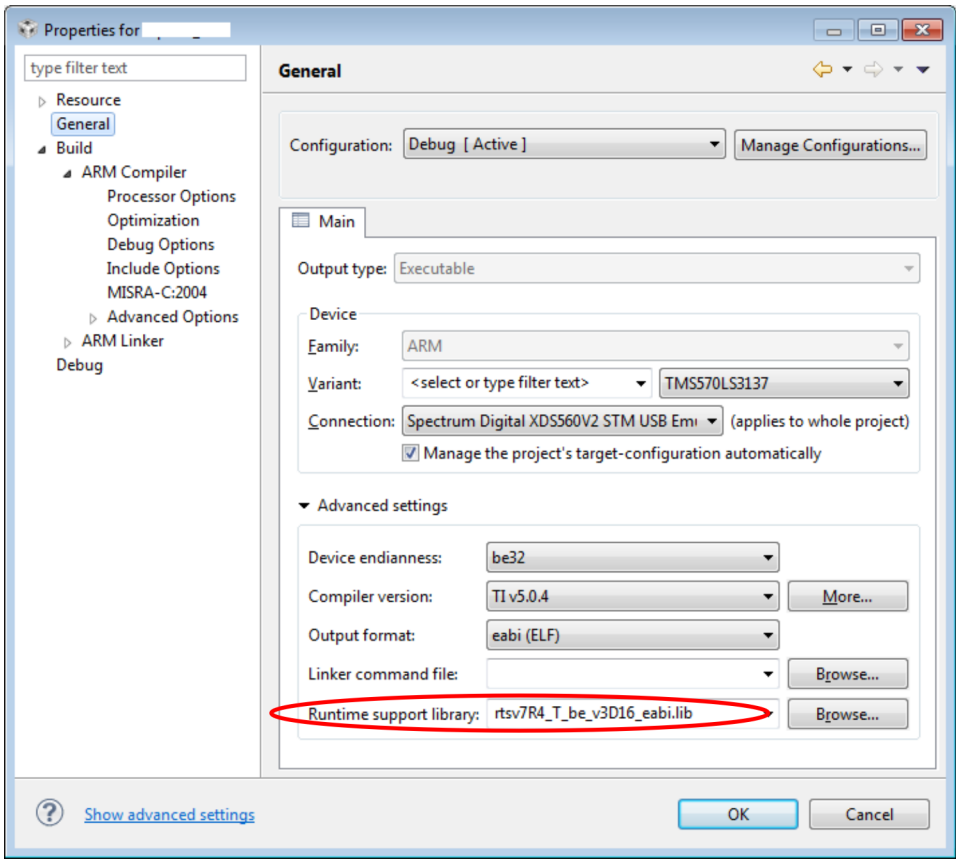

Figure 13. Runtime Library Select

10 How to Add Your Test Cases?

1. Add a new folder, For Example *MyTests* in the Test folder created by the tool.
2. Define the test sequence as shown in Figure 14 and save it in the folder; for example *Test1* inside *MyTests*.
3. Add more test sequences by adding more folders in *MyTests*. Figure 14 shows a template of a test sequence.

	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 14. Test Sequence Template

Strictly follow the folder structure shown in Figure 15.

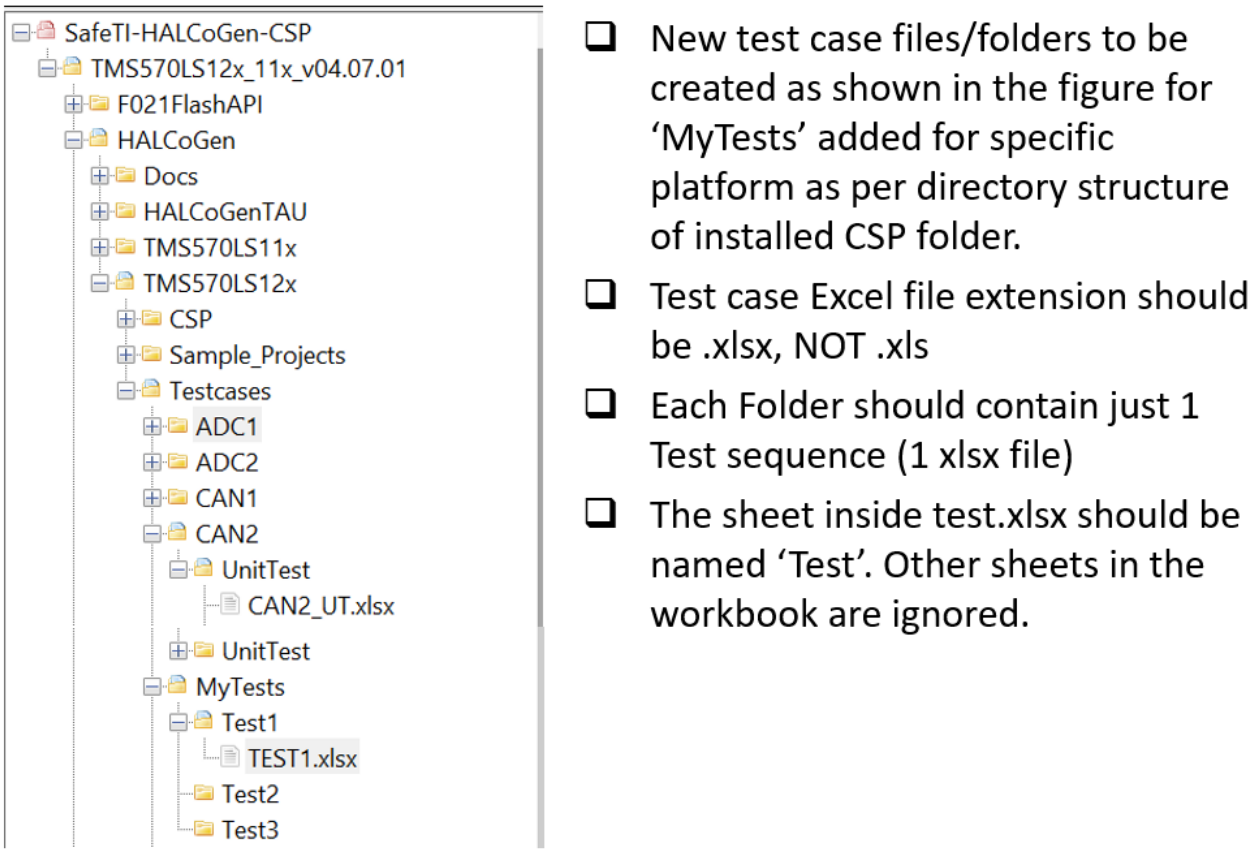

Figure 15. Folder Structure

Points to keep in mind while writing a test sequence:

- The test sequence must begin with the "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, 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 (that includes the function input parameters, return value, global variables checked) must be defined for each test case as shown in Figure 16 Df_type information is used by LDRA tool to understand the type of input and output parameters, for example if the parameter is a global or local variable. Commonly used Df_Types are
  - "**Z**" - input parameter through Local
  - "**H**" - output Global
  - "**%**" - Function return
  - "**O**" - output parameter
  - "**G**" - input parameter through Global
- The file under test must be defined for each test case and the function tested must be defined in that file.
- Each test cases must be separated by a blank row.
- End of test sequence must be written in column 3 after the last test case leaving a blank row.
- One test sequence can test one and only one file.
- If the file under test has to be tested in white box mode, the Excel sheet must be saved as *_UT.xlsx (UT stands for Unit Test). In this case, a Code coverage report will also be generated along with the regression report.
- If the file under test has to be tested in black box mode, the Excel sheet must be saved as *_FT.xlsx (FT stands for functional test).

Figure 16 shows a few examples of a test case.

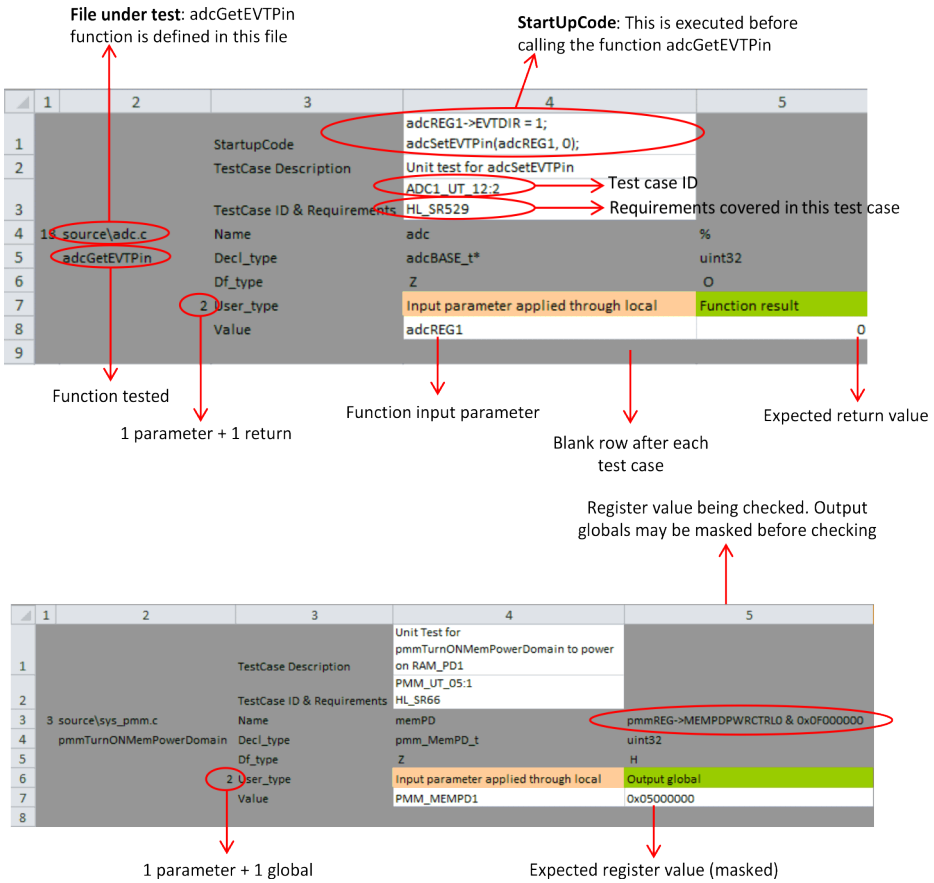

Figure 16. Examples of a Test Case

NOTE: Clicking the *Add testcases* button replaces all of the predefined test folders. It s recommended to add new test folders rather than modifying the existing ones.

Useful Tip: Right-click on the test list field and press *Refresh* to regenerate the list.

11 Reports

The following are reports generated at the end of the 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. It does not keep record of the previous test runs.

11.2 Dynamic Coverage Analysis Report

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

- Statement coverage
- Branch and decision coverage
- MC/DC coverage (modified condition and decision coverage)

Unlike the regression report, this is an accumulated report of all the previous runs.

12 FAQ

12.1 Why Does the Application Show “Code Generation Failed"?

In the HALCoGenTAU make sure HALCoGen Directory points to HALCoGen version 4.00.00 or greater.

12.2 Why Does the TCF Generation Fail?

- The test sequence is open in Microsoft Excel
- TCF generation may get stuck if a test case (including the last one) is not terminated by a blank row
- TCF generation may get stuck if the keyword “End of Test Sequence” is not found in the Excel sheet

12.3 What are the Possible Reasons for Build Failure?

- License initialization failure
  - Check whether the LDRA license is properly installed or expired.
- Validation failure
  - File name defined is wrong or the function is not found in the defined file name
  - The number of parameters for the test case defined in the Excel does not match the actual value. Note that number of parameters includes function input parameters, function return and the global variables (or peripheral registers) checked.
- Previous analysis was terminated before the operation was completed
  - Delete the analysis folder present in LDRA Application data directory
- Compile error
  - For details, users can check the *CompileLog.txt* in the *<test folder>\Debug* folder.
- Invalid entries in build options file

12.4 From Where Do I See the Instrumented Code?

The Instrumented code is saved in the *<HALCoGen Project Dir>/LDRABackup/<TestName>* folder.

12.5 Is the Source Code Backed Up Before LDRA Instruments the Code During TAU Execution?

Yes, During compilation phase the source code is backed up temporarily in the *<HALCoGen Project Dir>/source/Ldra~bkp* folder. HALCoGen TAU restores the source file to the *<HALCoGen Project Dir>/source* folder once the compilation is complete and deletes Ldra~bkp folder. Due to system crash or application process being killed during the compilation phase, the original source file must be restored from the *<HALCoGen Project Dir>/source/Ldra~bkp* folder.

12.6 How do I Include a Folder or a Library in my Project?

Compiler Options specified in the Build Options file can be modified to include the folder (Add -i "<folder name>" in the compiler options).

12.7 Is the Tool Compatible Only With HALCoGen v04.02.00? Can I Test a Project Generated With an Older Version?

This tool is compatible with any production version of HALCoGen starting from version 3.06.00. But the test sequences provided along with this tool may fail. This is because there are some functions newly defined from v04.02.00 and that would cause a compile error in the test sequence. Users can remove those test cases or define new test sequences.

Also, HALCoGen v04.02.00 generates an additional header file, which defines few macros that are used as inputs to some of the functional tests. Either define the macro manually or redefine the functional test.

12.8 Why are the Functional Tests Disabled by Default?

Not all the functional test cases can be run with a particular HALCoGen configuration. Users can choose the tests applicable for their configuration.

12.9 How Do I Know Which Functional Tests are Applicable for my Configuration?

It is defined in the test case documentation provided with this tool.

12.10 Does the Tool Optimize the HALCoGen Code While Compiling?

It doesn't by default. Optimization can be enabled through build options file (for example, add –o2 in the compiler options)

12.11 It Says Test Execution Completed. But I Cannot Find the Report in the Reports Folder.

This can happen 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 before operation was completed

12.12 Test Execution Hung, 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 that.

This can happen if the test is stuck in some loop mainly because of wrong configuration of wrong selection of functional test.

12.13 Can I Use HALCoGen TAU to Test Source File Other Than HALCoGen Generated Files?

No. HALCoGen TAU which is part of HALCoGen-CSP is licensed to test HALCoGen generated files and support files only.

12.14 I Have Full Version of LDRA Tool Suite Instead of LDRAUnit. Can I Use This Tool With This License?

Currently HALCoGen TAU supports only LDRAUnit.

12.15 How to Get LDRA Unit and Setting up the License After Downloading HALCoGen-CSP From TI?

After downloading HALCoGen-CSP, 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 using the following steps:

1. After downloading HALCoGen-CSP LDRA-Less package, the customer can contact LDRA and acquire the LDRAunit-TI-Qual 9.4.3 (or higher).
2. 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
3. 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