



BTC EmbeddedPlatform - Tutorial

Formal Test and Requirements-based Test Generation with BTC EmbeddedPlatform

Version 2.5



1. About Formal Test
2. About Requirements-based Automatic Test Case Generation
3. Scenario
4. Prerequisites and additional information
5. Prepare the tutorial and start BTC EmbeddedPlatform
6. Import Test Cases
7. Execute Test Cases
8. Formal Test Execution Result
9. Creating Test Cases automatically
10. Repeat Test Cases Execution
11. Reporting

What is a Formal Test?

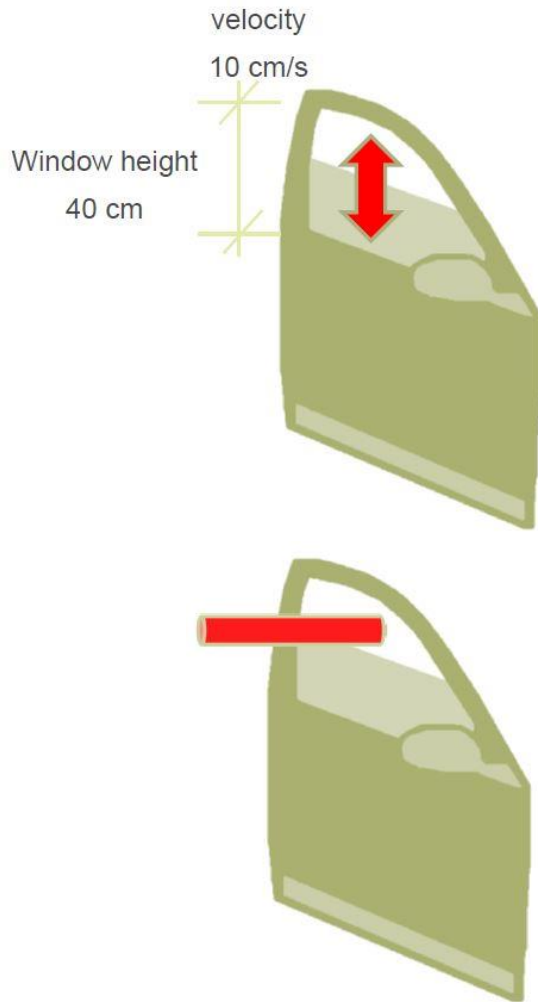
Formal Testing is a method that uses existing simulation results to verify if they fulfill or violate one or more formalized requirements.

What is the goal of a Formal Test?

The goal of this approach is to clarify whether the System under Test (SUT) behaves correctly as specified in the (formalized) requirements. Therefore the SUT will be stimulated with test cases that have been created either manually or automatically or even coming from a HIL simulation. The output of the SUT will be verified against a formalized requirement to see whether the requirement is fulfilled or violated by the data stream. If a correctly formalized requirement is violated this indicates that there is a misbehavior in the model that should be fixed.

This tutorial describes a standard workflow for Formal Test with BTC EmbeddedPlatform to prove the correctness of the model regarding a requirement. Regarding the formalization of requirements please refer to the corresponding tutorial.

Scenario 1/2



For this tutorial we will have a look at one requirement of the demo model „Power Window Controller“ which is a controller for the passenger side window.

Here are some key features of the controller:

- The window can be controlled from both, the passengers and the drivers side
- A driver command overrules a passenger command
- A tap function is provided to open or close the window completely if the switch is pressed for less than 1 second.
- It also provides an obstacle detection with the following properties:
 - If an obstacle is detected the window moves down immediately for 10 cm
 - Independent from current driver and passenger requests
 - Independent from active tap function

On the next slide you can find a list of requirements regarding the Power Window Controller.

Scenario (2/2)

REQ_ID	Description
REQ_PW_1_1	If the driver up switch is pressed, the window has to start moving up within 50 [ms].
REQ_PW_1_2	If the driver down switch is pressed, the window has to start moving down within 50 [ms].
REQ_PW_2_1	If the driver up or the passenger up switch is pressed for at most auto_up_time, the auto-up mode is activated and the window continues to move up.
REQ_PW_2_2	If the driver down or the passenger down switch is pressed for at most auto_down_time, the auto-down mode is activated and the window continues to move down.
REQ_PW_3	The driver commands have priority over the passenger commands.
REQ_PW_4_1	If an obstacle is detected, the window has to start moving down within 10 [ms].
REQ_PW_4_2	When an obstacle is detected, the window has to move down for emergency_down_time or until the bottom end is reached.

Prerequisites and additional information

To be able to perform a Formal Test in BTC EmbeddedPlatform, the requirement first has to be transformed into a formal representation.

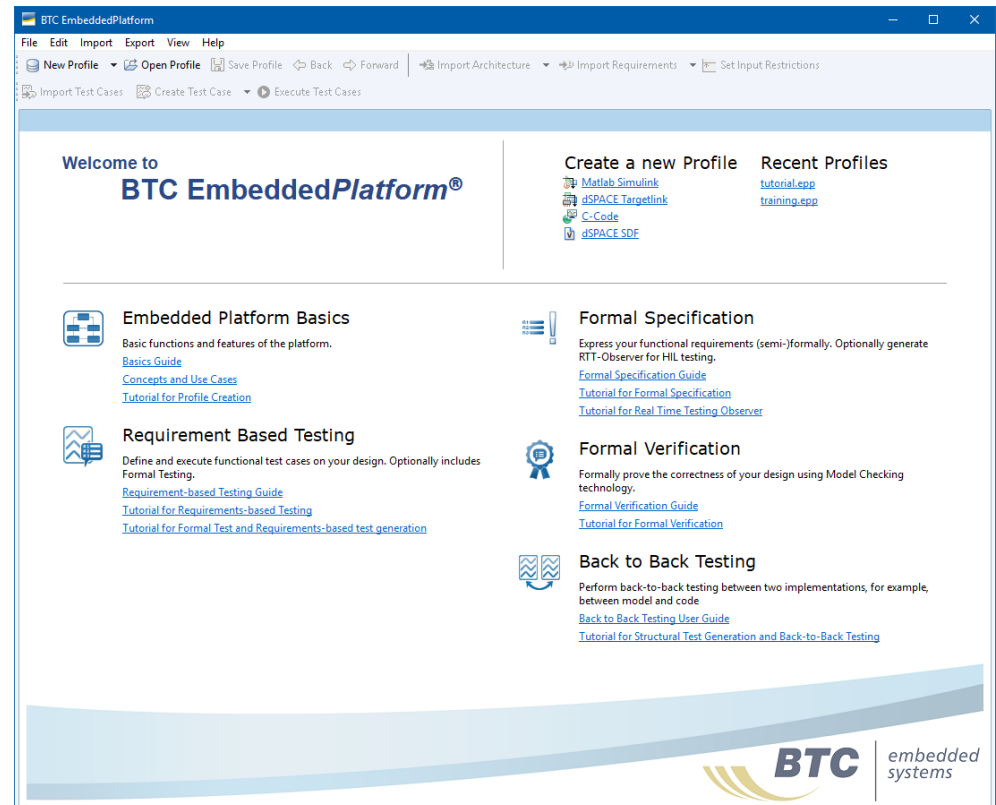
In other words, the requirement itself has to be unambiguous and machine-readable. BTC EmbeddedPlatform offers different capabilities that allow the transformation of an informal specification into a formal one by enriching it with a clear syntax and semantic.

Please copy the demo folder from the installation directory to any other directory that allows to modify files without administrator rights.

To start with this tutorial, you need to create a profile. If not already done, please refer to the “Tutorial for Formal Specification.pdf” tutorial to create a profile and formalize requirements. Please use the powerwindow_tl_v03.mdl for the architecture import.

Prepare the tutorial and start BTC EmbeddedPlatform

- Start BTC EmbeddedPlatform from the Windows Start Menu, Matlab or the Desktop shortcut if available. Once BTC EmbeddedPlatform has started, please get an overview of the tool.
- BTC EmbeddedPlatform provides several use cases.
 - Requirements-based Testing
 - Back-to-Back Testing
 - Formal Specification
 - Formal Verification
- All use cases are based on the same data structure and therefore can exchange information and data between the use cases.
- Each use case provides an own view on the data fitting to its workflow.



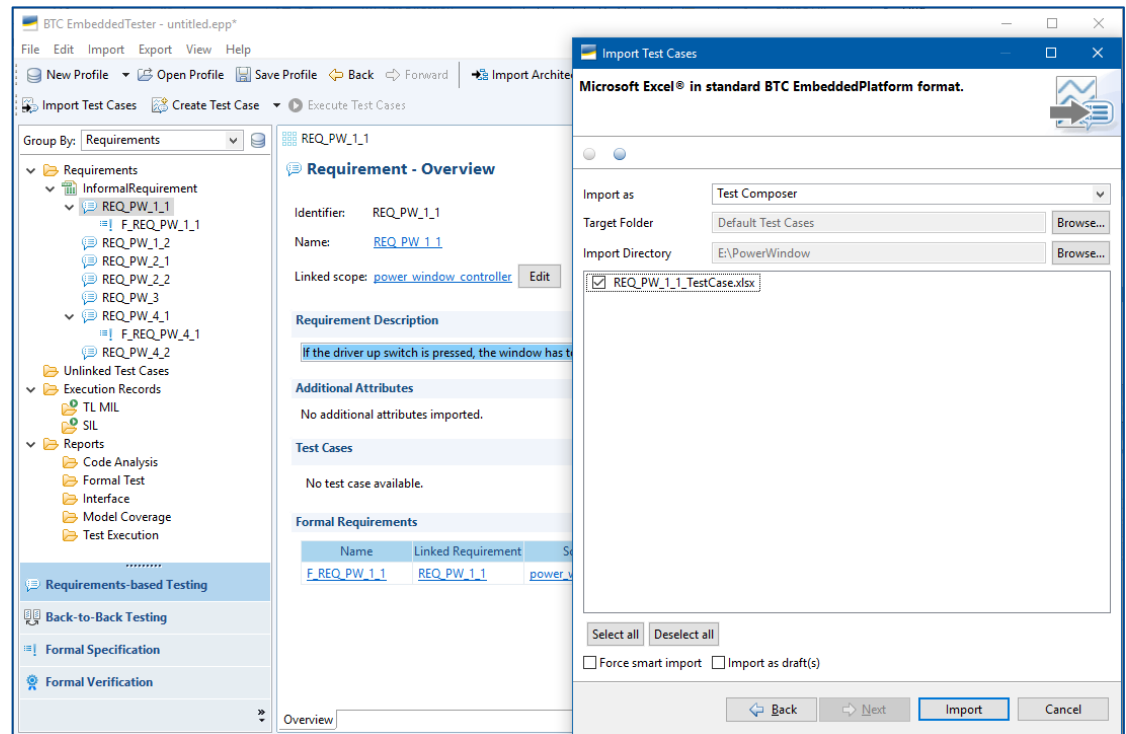
Import Test Cases

As already mentioned, the goal of a Formal Test is to check existing simulation runs against (formal) requirements. Therefore we will now import a test case that shall be used to test the first requirement (REQ_PW_1_1).

Please, select the Requirements-based Testing perspective, if not already done, and then click on the “Import Test Cases” button in the toolbar.

In the upcoming dialog, “Excel” is selected as default import format. Click Next.

Please, navigate to the import directory that contains the test case “REQ_PW_1_1_TestCase.xlsx” and select this test case below.



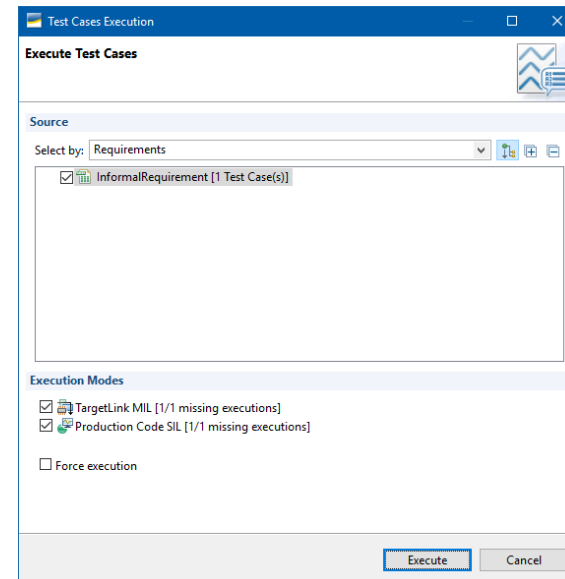
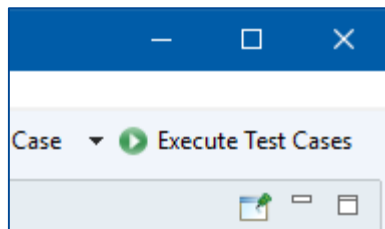
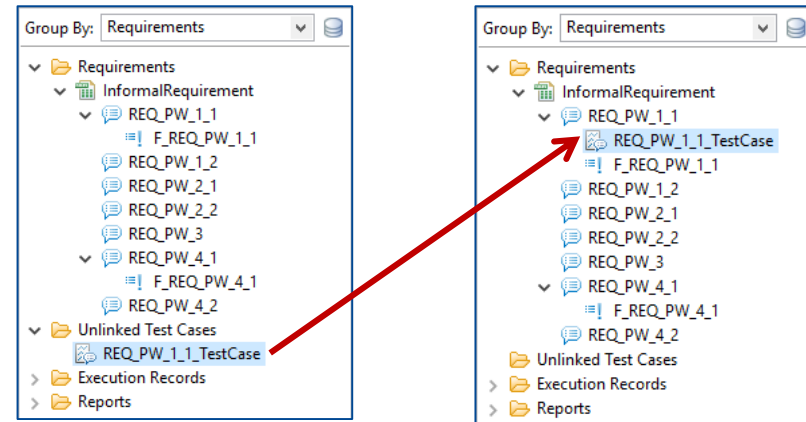
Press the “Import” button to complete this step.

Execute Test Cases

By default test cases are stored inside the “Unlinked Test Cases” folder. As the imported test case has been written for requirement number 1, we can drag and drop it onto the requirement. This way we ensure, that the test case is linked to the desired requirement.

To execute the test cases press the “Execute Test Cases” button in the toolbar. Select “Requirements” from the drop down in the upcoming dialog and both “Execution Modes”. Please press the “Execute” button to execute the test cases.

The Formal Test will be executed automatically once the execution records are available.



Formal Test Execution Result

Requirement Source - Requirements						
Name	Source	Number of Test Cases	Test Status TL MIL	Test Status SIL	FR Status SIL	FR Coverage SIL (ONCE)
REQ_PW_1_1	InformalRequirement	1	Passed	Passed	Fulfilled	Covered
REQ_PW_1_2	InformalRequirement	0	No linked Test Cases	No linked Test Cases	No linked Formal Requ...	No linked Formal Require...
REQ_PW_2_1	InformalRequirement	0	No linked Test Cases	No linked Test Cases	No linked Formal Requ...	No linked Formal Require...
REQ_PW_2_2	InformalRequirement	0	No linked Test Cases	No linked Test Cases	No linked Formal Requ...	No linked Formal Require...
REQ_PW_3	InformalRequirement	0	No linked Test Cases	No linked Test Cases	No linked Formal Requ...	No linked Formal Require...
REQ_PW_4_1	InformalRequirement	0	No linked Test Cases	No linked Test Cases	Inconclusive	Not covered
REQ_PW_4_2	InformalRequirement	0	No linked Test Cases	No linked Test Cases	No linked Formal Requ...	No linked Formal Require...

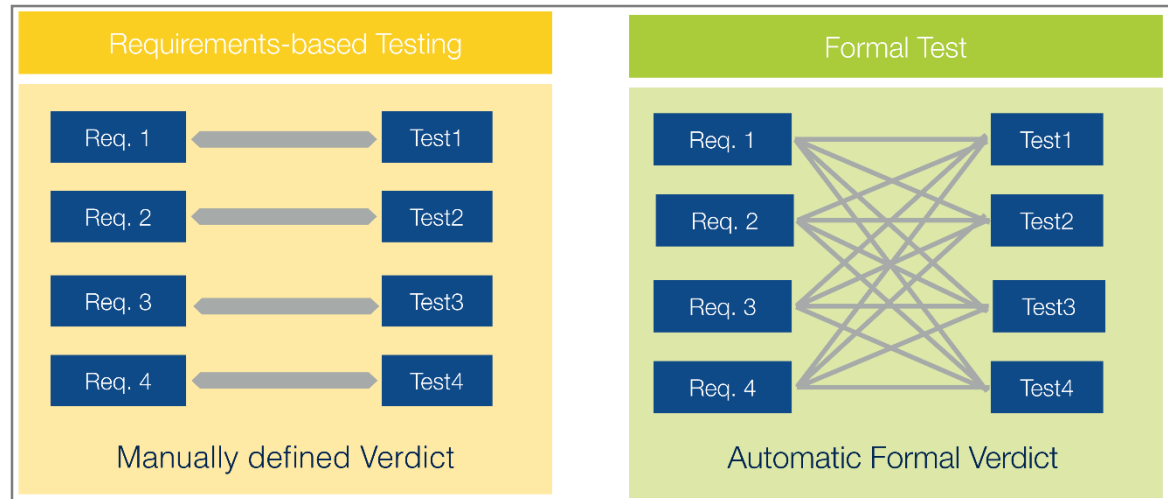
The results of the simulation will be visible inside the “Requirements Source” dashboard. As displayed on the left, the test case indeed covers requirement No. 1 and the test status is “Passed”.

Moreover it is visible that the test case has also influenced the analysis result for the second formal requirement (REQ_PW_4_1) even though the test case doesn't belong to this requirement.

This cross-checking mechanism which tests all test cases against all requirements is an important feature of the Formal Test and an advantage compared to requirements-based testing, as it is able to detect side effects on the one hand and to increase the depth of the test process on the other hand.

The Test Status of the second requirement is “Inconclusive” in this case as neither the trigger nor the action condition have been fulfilled by the imported test case.

This means, that REQ_PW_4_1 is not violated by our test case but it's also not yet tested.



About Requirement-based Automatic Test Case Generation

What is Requirement-based Automatic Test Case Generation

In contrast to the traditional approach, where test cases are created manually including stimulus and expected system behavior, the automated approach generates test cases based on formal requirements for both stimulus and expected values.

What is the goal of Requirement-based Automatic Test Case Generation?

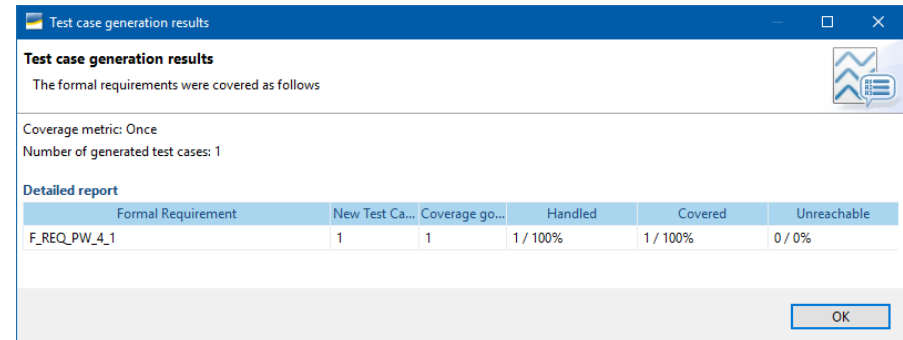
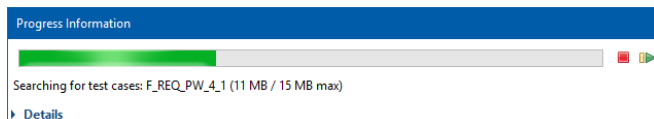
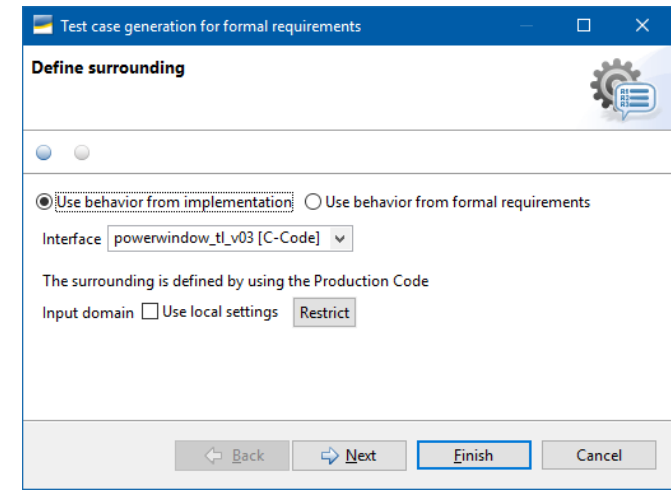
The goal of this approach is to find test cases that cover certain coverage goals, e.g. an action end event using model checking technologies.

The second part of this tutorial shows how to generate test cases from a formal requirement.

Creating Test Cases automatically

In order to cover the second requirement we could either import or create a second test case manually or we can let BTC EmbeddedPlatform create test cases automatically.

- Right-click on “REQ_PW_4_1” and choose “Generate Test Cases” from the context menu.
- In the upcoming user dialog, please select “Use behavior from implementation” in order to use the systems behavior as the intended behavior.
- Press the “Finish” button in order to start the test case generation process. A progress bar will indicate the current status.
- As soon as the execution has finished a Graphical User Interface will occur which summarizes the found vector(s).



Repeat Test Cases Execution

A new test case is generated that fulfills the requirement. However, the analysis of the requirements is based on the execution records and not on the test case itself. In order to affect the test result, the test cases have to be executed a second time using the same approach as before. This time also the newly created test case will be taken into account during the simulation.

As seen below the Test Status as well as the Formal Requirement Status and the Formal Requirement Coverage have changed. Now both requirements are covered by the available test cases and both tests are passed.

Requirement Source - Requirements						
Name	Source	Number of Test Cases	Test Status TL MIL	Test Status SIL	FR Status SIL	FR Coverage SIL (ONCE)
REQ_PW_1_1	InformalRequirement	1	Passed	Passed	Fulfilled	Covered
REQ_PW_1_2	InformalRequirement	0	No linked Test Cases	No linked Test Cases	No linked Formal Requ...	No linked Formal Require...
REQ_PW_2_1	InformalRequirement	0	No linked Test Cases	No linked Test Cases	No linked Formal Requ...	No linked Formal Require...
REQ_PW_2_2	InformalRequirement	0	No linked Test Cases	No linked Test Cases	No linked Formal Requ...	No linked Formal Require...
REQ_PW_3	InformalRequirement	0	No linked Test Cases	No linked Test Cases	No linked Formal Requ...	No linked Formal Require...
REQ_PW_4_1	InformalRequirement	1	Passed	Passed	Fulfilled	Covered
REQ_PW_4_2	InformalRequirement	0	No linked Test Cases	No linked Test Cases	No linked Formal Requ...	No linked Formal Require...

Even though the tests have been successful so far, this does not mean that the requirements can never be violated. It only means that there are currently no test cases (or to be precise execution records) available that violate the requirements. To be sure whether or not a requirement can be violated at all, BTC EmbeddedPlatform can be used with its use case Formal Verification.

For more information regarding this use case, please refer to the “Tutorial for Formal Verification.pdf”.

During testing, the dashboards show the current status of the test project, related to the selected item in the Profile Navigator. In addition it is sometimes necessary to export these information for documentation of the current status or presenting the test status to the customer. For Requirements-based testing with Formal Test BTC EmbeddedPlatform provides four different reports.

The **Test Execution Report** shows the simulation results of one architecture, e.g. TL_MIL or SIL. If this report is created on a requirements source, it does contain the Requirements Coverage in addition.

The **Formal Test Report** shows the simulation results of one architecture, e.g. TL_MIL and its coverage of the formalized requirements.

The **Code Analysis Report** contains the code coverage, based on the test cases in the profile, for each subsystem for all test goals on code level, e.g. Statement, Decision, Condition and MC/DC coverage.

The **Model Coverage Report** shows the model coverage, based on the test cases in the profile, for each subsystem for all test goals on model level, e.g. Statement, Decision, Condition and MC/DC coverage. Please note: To create a Model Coverage Report, you need to have a Simulink Verification and Validation toolbox license.

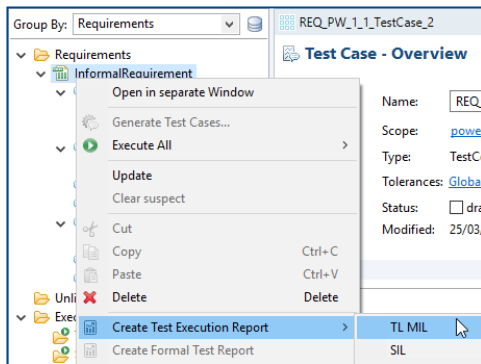
The **Interface Report** shows all interface objects that are part of the system under test and lists them for each Scope separately.

A report can be generated based on a Requirements Source, Test Case folder and a Scope.

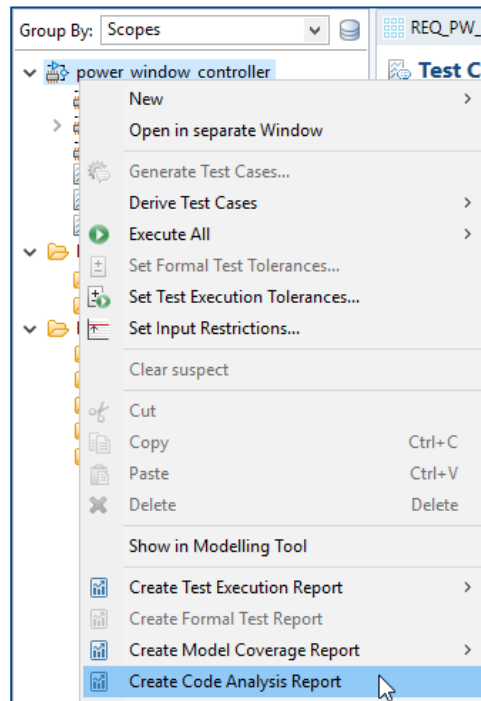
Reporting (2/7)

To create a report, right-click on one of the items in the Profile Navigator and select the desired report from the context menu.

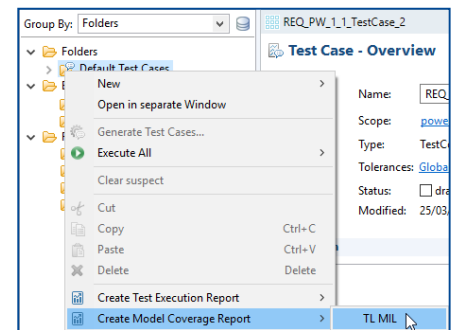
Test Execution Report



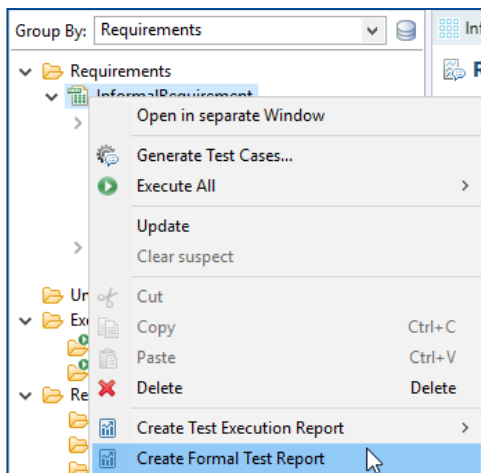
Code Coverage Report



Model Coverage Report

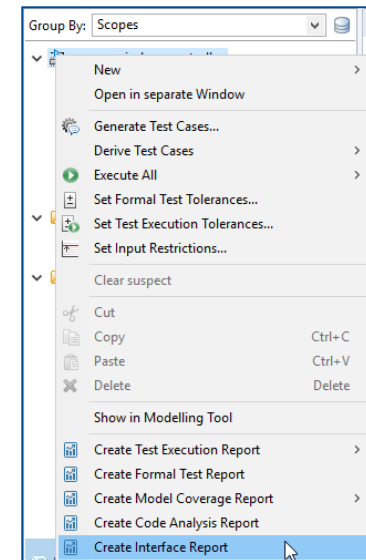


Formal Test Report

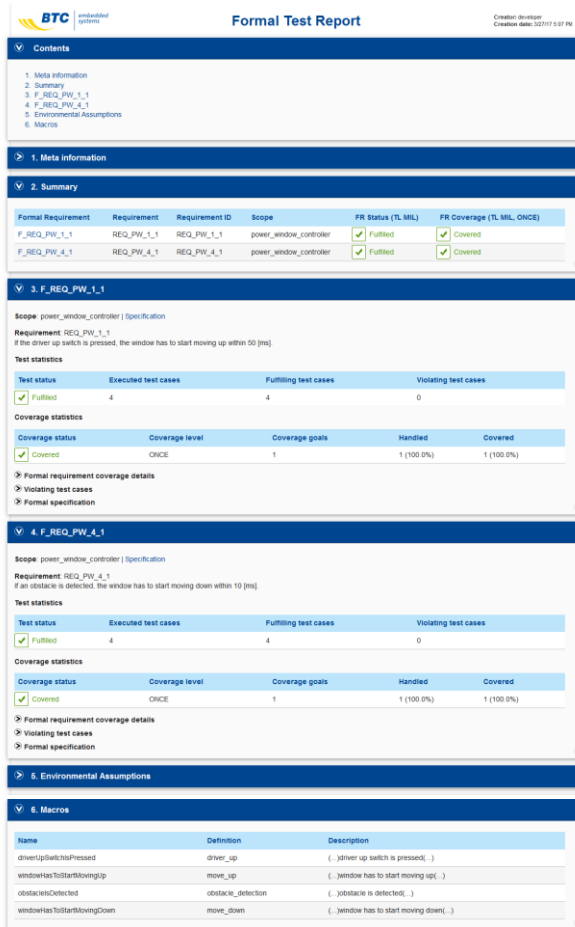


Once a report is created, it is listed in the Profile Navigator as a new item. The report can be exported with a right-click on the item and select “Export Report ...” from the context menu.

Interface Report



Formal Test Report



The screenshot displays the 'Formal Test Report' interface. It features a sidebar with a 'Contents' menu and a main content area. The main content area shows a table of requirements and their coverage. The table has columns for 'Formal Requirement', 'Requirement', 'Requirement ID', 'Scope', 'FR Status (TL MIL)', and 'FR Coverage (TL MIL, ONCE)'. The requirements listed are 'F_REQ_PW_1_1' and 'F_REQ_PW_4_1', both with a status of 'Fulfilled' and coverage of 'Covered'. Below the table, there are sections for '3. F_REQ_PW_1_1' and '4. F_REQ_PW_4_1', each containing a 'Test statistics' table and a 'Coverage statistics' table. The 'Test statistics' tables show 'Test status' (Fulfilled), 'Executed test cases' (4), 'Fulfilling test cases' (4), and 'Violating test cases' (0). The 'Coverage statistics' tables show 'Coverage status' (Covered), 'Coverage level' (ONCE), 'Coverage goals' (1), 'Handled' (1 (100.0%)), and 'Covered' (1 (100.0%)).

Formal Requirement	Requirement	Requirement ID	Scope	FR Status (TL MIL)	FR Coverage (TL MIL, ONCE)
F_REQ_PW_1_1	REQ_PW_1_1	REQ_PW_1_1	power_window_controller	Fulfilled	Covered
F_REQ_PW_4_1	REQ_PW_4_1	REQ_PW_4_1	power_window_controller	Fulfilled	Covered

3. F_REQ_PW_1_1

Scope: power_window_controller | Specification

Requirement: REQ_PW_1_1
If the driver up switch is pressed, the window has to start moving up within 50 [ms].

Test statistics

Test status	Executed test cases	Fulfilling test cases	Violating test cases
Fulfilled	4	4	0

Coverage statistics

Coverage status	Coverage level	Coverage goals	Handled	Covered
Covered	ONCE	1	1 (100.0%)	1 (100.0%)

4. F_REQ_PW_4_1

Scope: power_window_controller | Specification

Requirement: REQ_PW_4_1
If an obstacle is detected, the window has to start moving down within 10 [ms].

Test statistics

Test status	Executed test cases	Fulfilling test cases	Violating test cases
Fulfilled	4	4	0

Coverage statistics

Coverage status	Coverage level	Coverage goals	Handled	Covered
Covered	ONCE	1	1 (100.0%)	1 (100.0%)

5. Environmental Assumptions

6. Macros

Name	Definition	Description
driverUpSwitchPressed	driver_up	(_driver up switch is pressed, _)
windowHasToStartMovingUp	move_up	(_window has to start moving up, _)
obstacleDetected	obstacle_detection	(_obstacle is detected, _)
windowHasToStartMovingDown	move_down	(_window has to start moving down, _)

The Formal Test Report is showing the requirements coverage of an simulation of test cases on a specific architecture, e.g. TL_MIL. Each section can be expanded or collapsed.

The available section are:

1. Meta information
2. Summary of available formal requirements
3. A detailed view for each requirement
4. Environmental Assumptions
5. Macros

Test Execution Report

BTC embedded systems		Test Execution Report	Creator: devision Creation date: 3/22/17 11:10 PM
Contents			
1. Meta Information			
2. Global Tolerances			
3. Summary			
4. Requirements Traceability			
5. Test Execution Results Overview			
6. Detailed Execution Results			

power_window_controller								
Check	Name	Kind	Data type	Resolution	Offset	Relative tolerance [%]	Absolute tolerance [unit]	Lead tolerance [s]
✓	detection_endstop_top	Local	Bool	1.0	0.0	0.0	0.0	0.0
✓	detection_obstacle	Local	Bool	1.0	0.0	0.0	0.0	0.0
✓	obstacle_detection	Output	Bool	1.0	0.0	0.0	0.0	0.0
✓	move_up	Output	Bool	1.0	0.0	0.0	0.0	0.0
✓	move_down	Output	Bool	1.0	0.0	0.0	0.0	0.0
✓	detection_endstop_bottom	Local	Bool	1.0	0.0	0.0	0.0	0.0

validate_passenger								
Check	Name	Kind	Data type	Resolution	Offset	Relative tolerance [%]	Absolute tolerance [unit]	Lead tolerance [s]
✓	validated_up	Output	Bool	1.0	0.0	0.0	0.0	0.0
✓	validated_down	Output	Bool	1.0	0.0	0.0	0.0	0.0
✓	validated_neutral	Output	Bool	1.0	0.0	0.0	0.0	0.0

3. Summary	
Number of Test Cases	2
Passed	2
Failed	0
Outdated	0
Error	0
Missing Executions	0
Execution Mode	SL

4. Requirements Traceability			
Requirement ID	Description	Test Cases	Covered
REQ_PW_1_1	If the driver up switch is pressed, the window has to start moving up within 50 [ms]	0	✗
REQ_PW_1_2	If the driver down switch is pressed, the window has to start moving down within 50 [ms]	1	✓
REQ_PW_3_1	If the driver up or the passenger up switch is pressed for at most auto_up_time, the auto-up mode is activated and the window continues to move up.	0	✗
REQ_PW_3_2	If the driver down or the passenger down switch is pressed for at most auto_down_time, the auto-down mode is activated and the window continues to move down.	0	✗
REQ_PW_3	The driver commands have priority over the passenger commands.	1	✓
REQ_PW_4_1	If an obstacle is detected, the window has to start moving down within 10 [ms]	0	✗
REQ_PW_4_2	When an obstacle is detected, the window has to move down for emergency_down_time or until the bottom end is reached	0	✗
Summary		2	20%

5. Test Execution Results Overview					
Test Case	Length	Requirement (report base)	Scope	Folder	Result
REQ_PW_1_2_TestCase_1	1	REQ_PW_1_2	validate_passenger	Default Test Cases	✓ Passed
REQ_PW_3_TestCase_1	11	REQ_PW_3	power_window_controller	Default Test Cases	✓ Passed

6. Detailed Execution Results	
✓	REQ_PW_1_2_TestCase_1 - Passed
✓	REQ_PW_3_TestCase_1 - Passed

The Test Execution Report is showing the result of an simulation of test cases on a specific architecture, e.g. TL_MIL. Each section can be expanded or collapsed.

The available section are:

1. Meta information
2. Global Tolerances
3. Summary
4. Requirements Traceability
5. Test Execution Results Overview
6. Detailed Execution Results

Code Analysis Report



Code Analysis Report
Creation: 2020-11-12 14:00
Creation Date: 2020-11-12 14:00

Table of Contents

- 1. Meta Information
- 2.1. detect_obstacle_endstop
- 2.2. validate_driver
- 2.3. validate_passenger

1. Meta Information

Scope path: powerwindow_v401power_window_controller/subsystempower_window_controller

Engine configurations

Children Scopes

powerwindow_v401power_window_controller/subsystempower_window_controller/detect_obstacle_endstop
powerwindow_v401power_window_controller/subsystempower_window_controller/validate_driver
powerwindow_v401power_window_controller/subsystempower_window_controller/validate_passenger

Coverage Goals

Statement (100)
Decision/branch (10)
Condition (25)
Condition/Decision (100)
Modified Condition/Decision (100)
Function (0)
Relational Operator (0)
Robustness Operator (10)

Robustness Checks Goals

Download (0)
Division By Zero (0)

Coverage Statistics

	Coverage Properties	Handled	Covered	Unreachable (ninf)
Statement Coverage	100	100	100.0%	0
Decision/branch Coverage	10	10	100.0%	0
Condition Coverage	25	25	100.0%	0
Condition/Decision Coverage	100	100	100.0%	0
Modified Condition/Decision	100	100	100.0%	0
Function Coverage	0	0	0.0%	0
Switch-Case Coverage	0	0	0.0%	0
Relational Operator Coverage	0	0	0.0%	0

Robustness Checks Statistics

	Robustness Check Properties	Handled	Covered	Unreachable (ninf)
Download	0	0	0.0%	0
Division By Zero	0	0	0.0%	0

Coverage Details

Robustness Checks Details

2.1. detect_obstacle_endstop

Scope path: powerwindow_v401power_window_controller/subsystempower_window_controller/detect_obstacle_endstop

Engine configurations

Children Scopes

Robustness Checks Goals

Download (0)
Division By Zero (0)

Coverage Statistics

	Coverage Properties	Handled	Covered	Unreachable (ninf)
Statement Coverage	5	5	100.0%	0
Decision/branch Coverage	0	0	0.0%	0
Condition Coverage	22	22	100.0%	0
Condition/Decision Coverage	15	15	100.0%	0
Modified Condition/Decision	1	1	100.0%	0
Function Coverage	0	0	0.0%	0
Switch-Case Coverage	0	0	0.0%	0
Relational Operator Coverage	0	0	0.0%	0

Robustness Checks Statistics

	Robustness Check Properties	Handled	Covered	Unreachable (ninf)
Download	0	0	0.0%	0
Division By Zero	0	0	0.0%	0

Coverage Details


Robustness Checks Details

The Code Analysis Report is showing the coverage of all available coverage goals on code level based on the current existing test cases in the profile. Each section of the report can be expanded or collapsed.

The available section are:

1. Meta information
2. A section for each subsystem
 - a) Scope path
 - b) Engine Configuration, if there are any
 - c) Overview of the coverage/robustness goals
 - d) Coverage/Robustness statistics
 - e) A detailed view of the coverage goals

Interface Report

**Interface Report**

Creator: developer
Creation date: 12/3/18 12:48 PM
Expand/Collapse All

☰ Table of Contents

1. Meta Information
2. power_window_controller
 2.1. detect_obstacle_endstop
 2.2. validate_driver
 2.3. validate_passenger

🔍 1. Meta Information

☰ 2. power_window_controller

Scope path: power_window_controller/Subsystem/power_window_controller

Interface Summary

🔍 Inputs 6 📏 Calibrations 4 🖥 Displays 3 📡 Outputs 3

☰ Imports

Name	Default Name	Data type	Resolution	Offset	Minimum	Maximum
driver_up	driver_up	Bool	1.0	0.0	0.0	1.0
driver_down	driver_down	Bool	1.0	0.0	0.0	1.0
passenger_up	passenger_up	Bool	1.0	0.0	0.0	1.0
passenger_down	passenger_down	Bool	1.0	0.0	0.0	1.0

■ ■ ■

☰ Outputs

Name	Default Name	Data type	Resolution	Offset	Minimum	Maximum
obstacle_detection	obstacle_detection	Bool	1.0	0.0	0.0	1.0
move_up	move_up	Bool	1.0	0.0	0.0	1.0
move_down	move_down	Bool	1.0	0.0	0.0	1.0

Children Scopes

power_window_controller/Subsystem/power_window_controller/detect_obstacle_endstop
power_window_controller/Subsystem/power_window_controller/validate_driver
power_window_controller/Subsystem/power_window_controller/validate_passenger

🔍 2.1. detect_obstacle_endstop

🔍 2.2. validate_driver

🔍 2.3. validate_passenger

The Interface Report gives an overview of all interface objects of each Scope in the profile. Beside the naming of the interface object this covers the data type, resolution, offset and minimum and maximum value ranges. Each section of the report can be expanded or collapsed.

The available section are:

1. Meta information
2. A section for each subsystem describing the interfaces

Model Coverage Report

Model Coverage Report

Results are based on Simulink® Verification and Validation Toolbox.

Global Statistics

Detailed Results

Global Statistics

Quick Links

General Information

Coverage Statistics

Hierarchical Statistics

powerwindow_it_v01/power_window_controller/Subsystem/power_window_controller

powerwindow_it_v01/power_window_controller/Subsystem/power_window_controller/detect_obstacle_endstop

powerwindow_it_v01/power_window_controller/Subsystem/power_window_controller/validate_driver

powerwindow_it_v01/power_window_controller/Subsystem/power_window_controller/validate_passenger

General Information

General Information

Simulation Mode

Generated for Scripts

Test Cases

Short-circuit Logic

Quick Links

Coverage Statistics

General Coverage

Simulink V&V Coverage

Decision Coverage

Condition Coverage

MC/DC Coverage

Transition Coverage

Simulink V&V Coverage

Transition Decision Coverage

Transition Condition Coverage

Transition MC/DC Coverage

State Coverage

Simulink V&V Coverage

State Coverage

Detailed Results

Quick Links

Hierarchical Statistics

Subsystem 1

powerwindow_it_v01/power_window_controller/Subsystem/power_window_controller

Coverage Statistics

Subsystem 2

powerwindow_it_v01/power_window_controller/Subsystem/power_window_controller/detect_obstacle_endstop

Simulink V&V Coverage

Decision Coverage

Condition Coverage

MC/DC Coverage

Detailed Results

Quick Links

Subsystem 3

powerwindow_it_v01/power_window_controller/Subsystem/power_window_controller/validate_driver

Simulink V&V Coverage

Decision Coverage

Condition Coverage

MC/DC Coverage

Detailed Results

Quick Links

Subsystem 4

powerwindow_it_v01/power_window_controller/Subsystem/power_window_controller/validate_passenger

Simulink V&V Coverage

Decision Coverage

Condition Coverage

MC/DC Coverage

Detailed Results

Quick Links

© STC Embedded Systems AG

The Model Coverage Report is showing the coverage of all available coverage goals on model level based on the current existing vectors in the profile. Each section of the report can be expanded or collapsed.

The available section are:

1. General Information
2. Coverage statistics for the top level and below for each subsystem



BTC Embedded Systems AG

Gerhard-Stalling-Straße 19, 26135 Oldenburg, GERMANY

Tel.: +49 441 969738 - 50

Fax: +49 441 969738 - 64

www.btc-es.de

