BTC EmbeddedPlatform - Tutorial

Formal Specification

Version 2.5

Content

About Formal Specification

What is Formal Specification?

Formal Specification means a computer-aided transformation of informal requirement notations into machine-readable, formal notations.

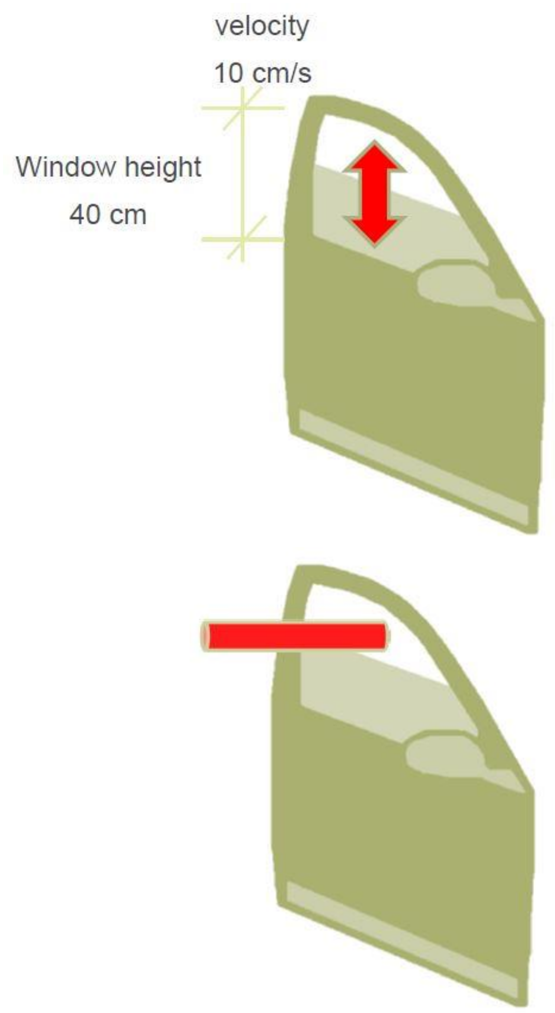
What is the goal of a Formal Specification?

The goal of Formal Specification is to enrich an informal specification with a clear syntax and semantic in order to improve its quality, make it unambiguous and to be able to automate further verification tasks.

As the creation of requirements stands at the beginning of any development process they can be seen as the basis for every further step in the development process. Therefore the quality and unambiguity of requirements is essential for the progress and success of each project – especially in safety critical environments.

The following tutorial describes a standard workflow for Formal Specification with BTC EmbeddedPlatform for a dSPACE TargetLink Model. All architecture-depended parts can also be applied to other architecture sources (Simulink model, C-Code) accordingly.

Scenario (1/2)



To become familiar with this tutorial, we will have a look at the requirements of the demo model „Power Window Controller“ which is a controller for the passenger side window of a vehicle.

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.

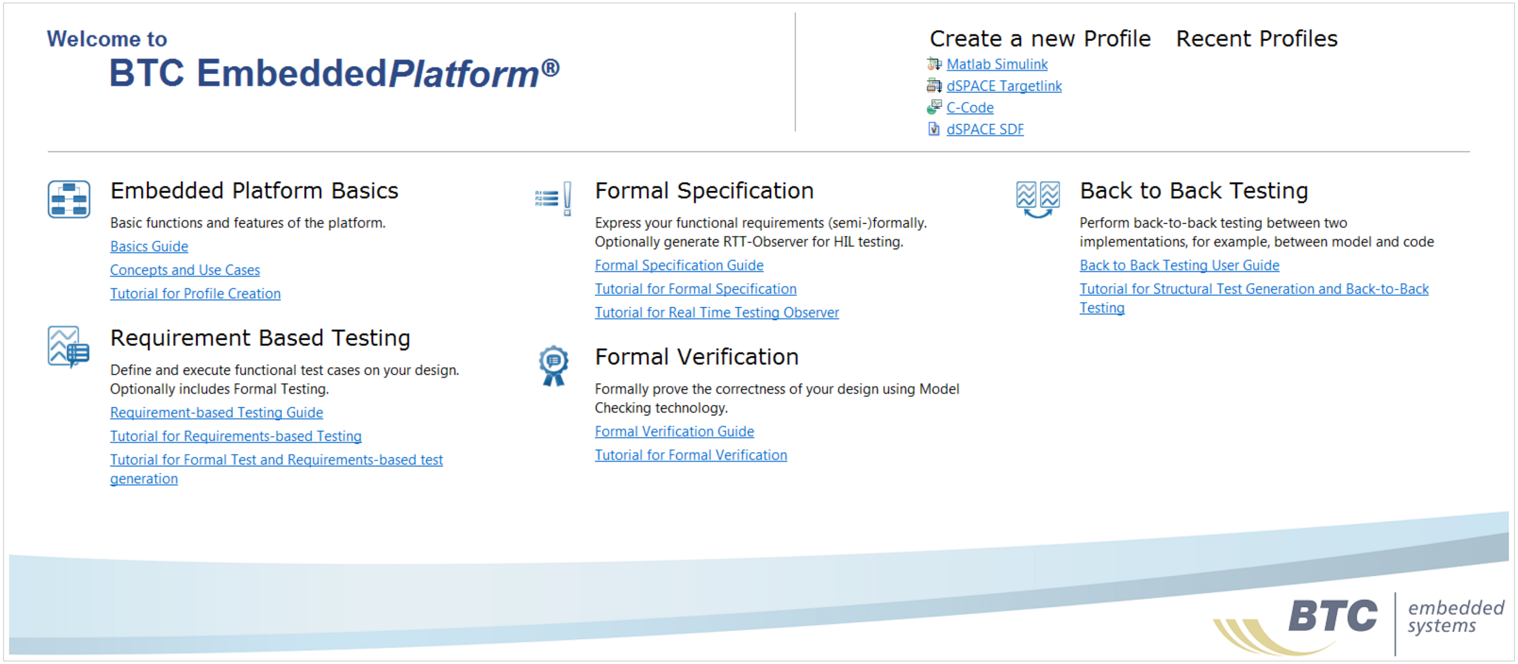
Scenario (2/2)

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.

BTC embedded systems

Start BTC EmbeddedPlatform

- To prepare the tutorial, please copy the folder with the demo model to a location with standard access rights and ensure that the “Read Only” attribute of the tutorials folder is not set after copying it. The location of the demo model usually is “C:\Program Files\BTC\<version>\tutorials\PowerWindow\PowerWindow_TargetLink”.
- Start BTC EmbeddedPlatform from the Windows Start Menu, Matlab or the Desktop shortcut if available. Once the BTC EmbeddedPlatform has started, please get an overview of the tool. The welcome screen shown below provides a good entry point for different topics.



Welcome to
BTC Embedded*Platform*®

Create a new Profile — Recent Profiles
- Matlab Simulink
- dSPACE Targetlink
- C-Code
- dSPACE SDF

Embedded Platform Basics
Basic functions and features of the platform.
- Basics Guide
- Concepts and Use Cases
- Tutorial for Profile Creation

Requirement Based Testing
Define and execute functional test cases on your design. Optionally includes Formal Testing.
- Requirement-based Testing Guide
- Tutorial for Requirements-based Testing
- Tutorial for Formal Test and Requirements-based test generation

Formal Specification
Express your functional requirements (semi-)formally. Optionally generate RTT-Observer for HIL testing.
- Formal Specification Guide
- Tutorial for Formal Specification
- Tutorial for Real Time Testing Observer

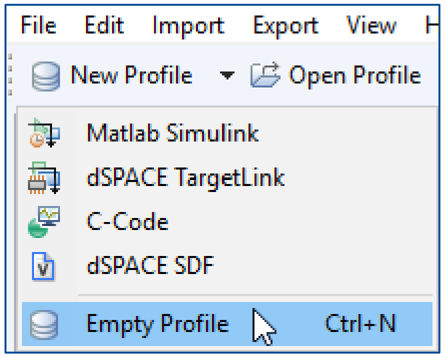
Formal Verification
Formally prove the correctness of your design using Model Checking technology.
- Formal Verification Guide
- Tutorial for Formal Verification

Back to Back Testing
Perform back-to-back testing between two implementations, for example, between model and code
- Back to Back Testing User Guide
- Tutorial for Structural Test Generation and Back-to-Back Testing

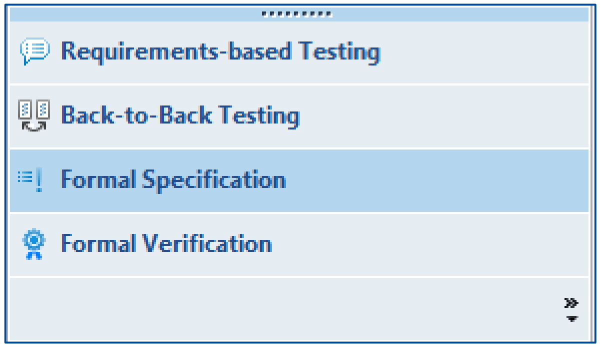
BTC embedded systems

Create a Profile

Once BTC EmbeddedPlatform has started, click on the “New Profile” button on the top left of the toolbar to create an empty profile. This triggers BTC EmbeddedPlatform to create a new profile which can be used for different use cases.



As the focus of this tutorial is Formal Specification, please switch to the “Formal Specification” use case first.

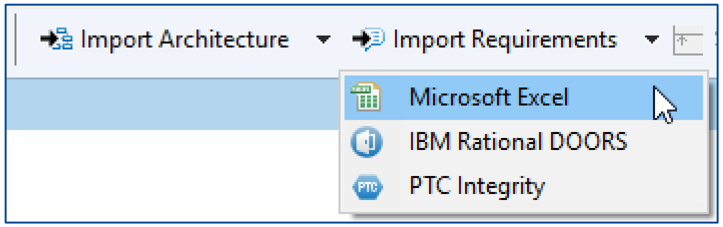


- All use cases are based on the same data structure and therefore can exchange information and data between the use cases.
- Each use case provides an own view on the data fitting to its workflow.

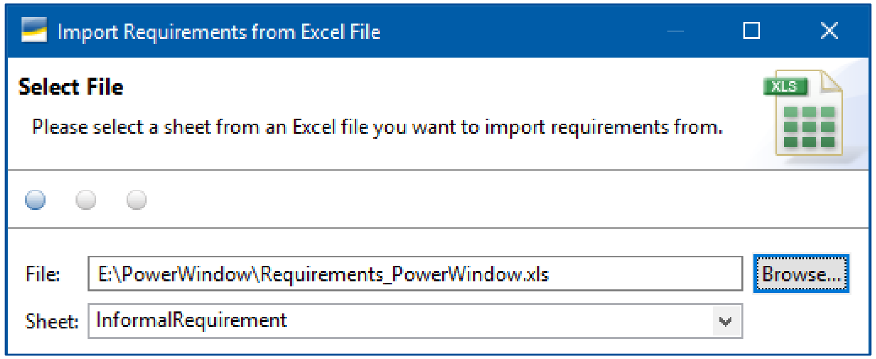
Import Requirements (1/3)

At the moment the profile is empty and there are no requirements present.

1. To import requirements to the profile, click on the arrow next to “Import Requirements” button and select Microsoft Excel from the upcoming menu.



2. Select the Excel file “*Requirements_PowerWindow.xls*” from “tutorials/PowerWindow” and press the “Next” button at the bottom of the dialog.



Import Requirements from Excel File

Select File

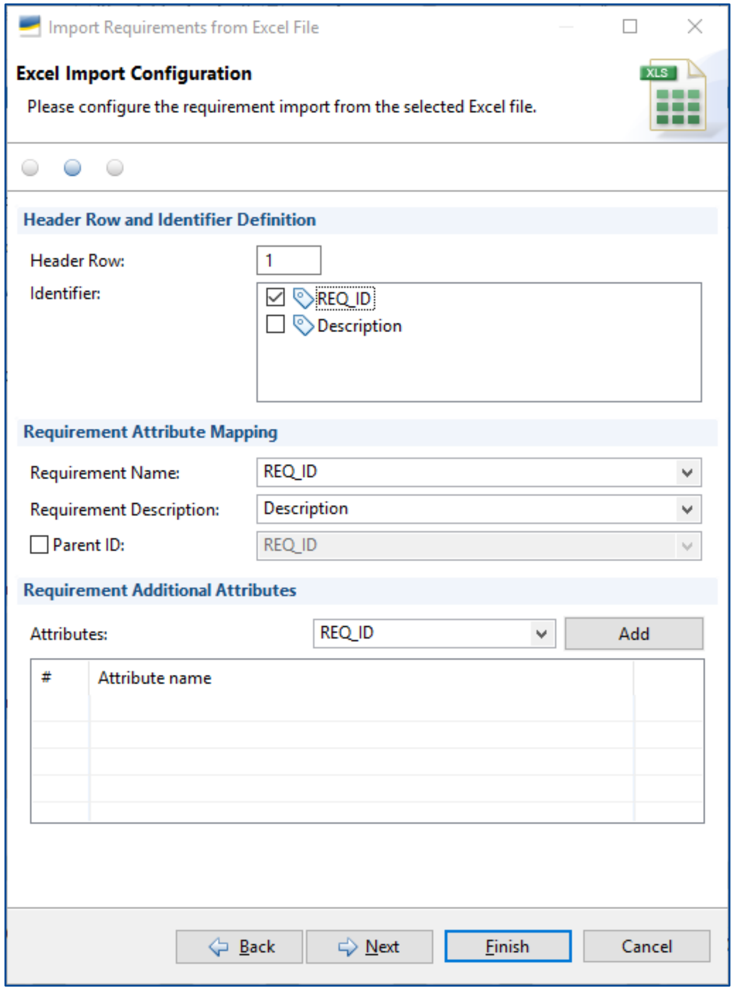
Please select a sheet from an Excel file you want to import requirements from.

File: E:\PowerWindow\Requirements_PowerWindow.xls [Browse...]

Sheet: InformalRequirement

Please note: It is also possible to import requirements from DOORS or PTC Integrity as well as customer specific formats.

Import Requirements (2/3)



Import Requirements from Excel File

Excel Import Configuration

Please configure the requirement import from the selected Excel file.

Header Row and Identifier Definition

Header Row: 1

Identifier: [✓ REQ_ID] Description

Requirement Attribute Mapping

Requirement Name: REQ_ID

Requirement Description: Description

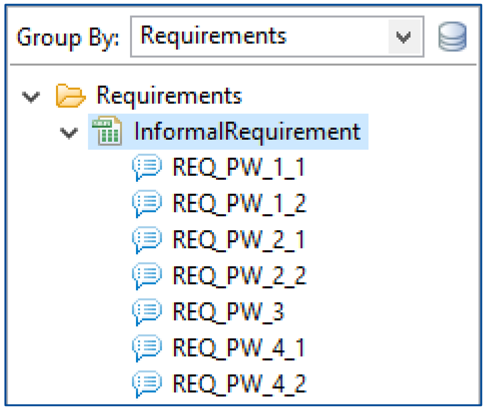
Parent ID: REQ_ID

Requirement Additional Attributes

Attributes: REQ_ID Add

#	Attribute name

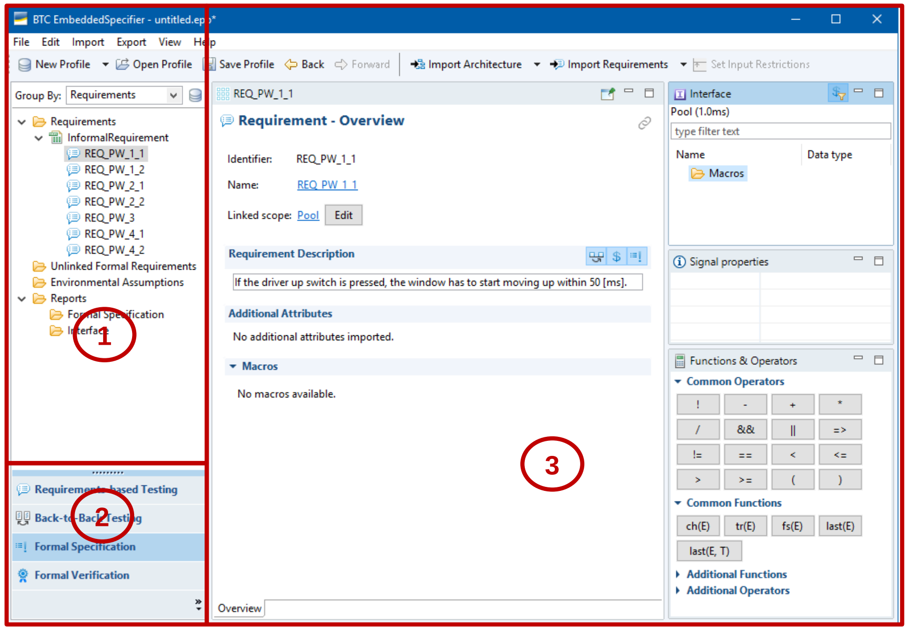
1. Select the header row (1)
2. Check the column which contains the unique identifier of the requirement (REQ_ID)
3. Map the attributes for Name and Description (in this case it automatically fits)
4. Press the “Finish” button.
5. The imported requirements are now present in the Profile Navigator. "Group By" selection displays "Requirements”.



Group By: Requirements

- Requirements
  - InformalRequirement
    - REQ_PW_1_1
    - REQ_PW_1_2
    - REQ_PW_2_1
    - REQ_PW_2_2
    - REQ_PW_3
    - REQ_PW_4_1
    - REQ_PW_4_2

Import Requirements (3/3)



Identifier: REQ_PW_1_1

Name: REQ_PW_1_1

Linked scope: Pool Edit

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

No additional attributes imported.

No macros available.

The Profile Navigator (1) shows the hierarchical structure of the imported requirements.

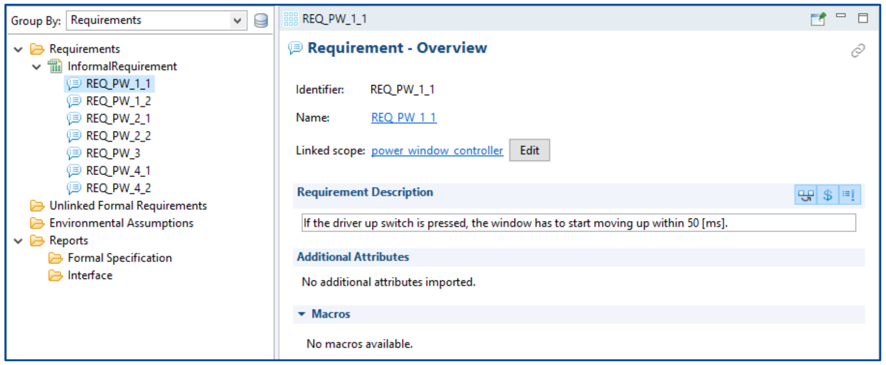
All artifacts that will be created in this tutorial will be connected to the referencing requirement.

The Use Case Selector (2) shows the provided use cases. Depending on your current license not all use cases might be available. In this tutorial you will focus on the Formal Specification use case.

The Dashboard (3) always shows the information corresponding to the selected item in the Profile Navigator. In this case the requirements description as well as some meta data are displayed in the center and additional information is provided on the right side with the Interface, Signal properties and Functions & Operator views.

Transformation: informal → semi-formal (1/6)

To be able to formalize an imported requirement, the first step is, to identify so-called "Macros".



Group By: Requirements

- Requirements
  - InformalRequirement
    - REQ_PW_1_1
    - REQ_PW_1_2
    - REQ_PW_2_1
    - REQ_PW_2_2
    - REQ_PW_3
    - REQ_PW_4_1
    - REQ_PW_4_2
  - Unlinked Formal Requirements
  - Environmental Assumptions
- Reports
  - Formal Specification
  - Interface

REQ_PW_1_1

Requirement - Overview

Identifier: REQ_PW_1_1

Name: REQ PW 1 1

Linked scope: power window controller Edit

Requirement Description

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

Additional Attributes

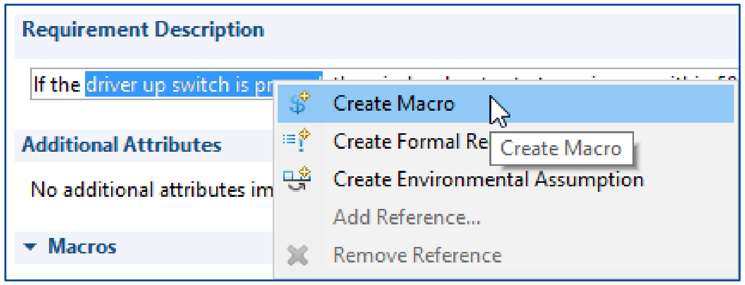
No additional attributes imported.

Macros

No macros available.

1. Click on the requirement "REQ_PW_1_1".
2. Mark the text part "driver up switch is pressed" in the "Description" field of the requirements editor.
3. Right-click on the marked part and select "Create Macro".
4. Repeat steps 2 & 3 for the text part "window has to start moving up" accordingly.

The selected parts will be transformed into macros which are also listed below.



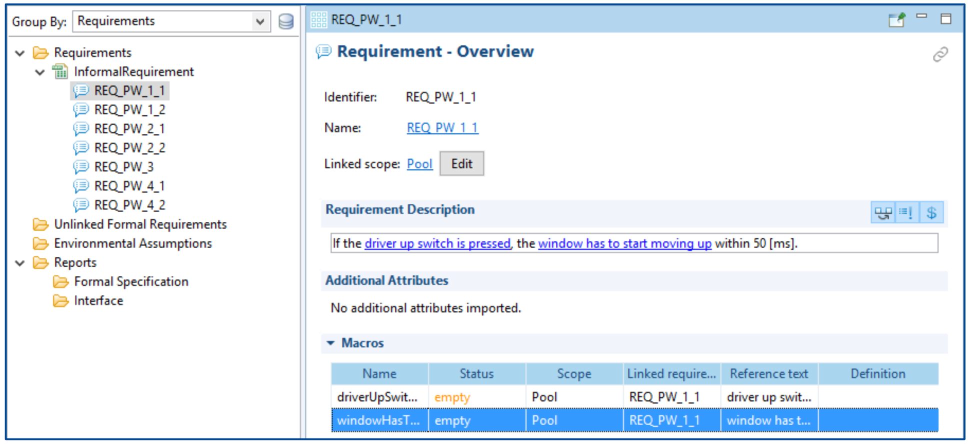
Requirement Description

If the driver up switch is pr

Additional Attributes

No additional attributes im

Macros



Group By: Requirements

- Requirements
  - InformalRequirement
    - REQ_PW_1_1
    - REQ_PW_1_2
    - REQ_PW_2_1
    - REQ_PW_2_2
    - REQ_PW_3
    - REQ_PW_4_1
    - REQ_PW_4_2
  - Unlinked Formal Requirements
  - Environmental Assumptions
- Reports
  - Formal Specification
  - Interface

REQ_PW_1_1

Requirement - Overview

Identifier: REQ_PW_1_1

Name: REQ PW 1 1

Linked scope: Pool Edit

Requirement Description

If the <u>driver up switch is pressed</u>, the <u>window has to start moving up</u> within 50 [ms].

Additional Attributes

No additional attributes imported.

Macros

Name	Status	Scope	Linked require...	Reference text	Definition
driverUpSwit...	empty	Pool	REQ_PW_1_1	driver up swit...	
windowHasT...	empty	Pool	REQ_PW_1_1	window has t...	

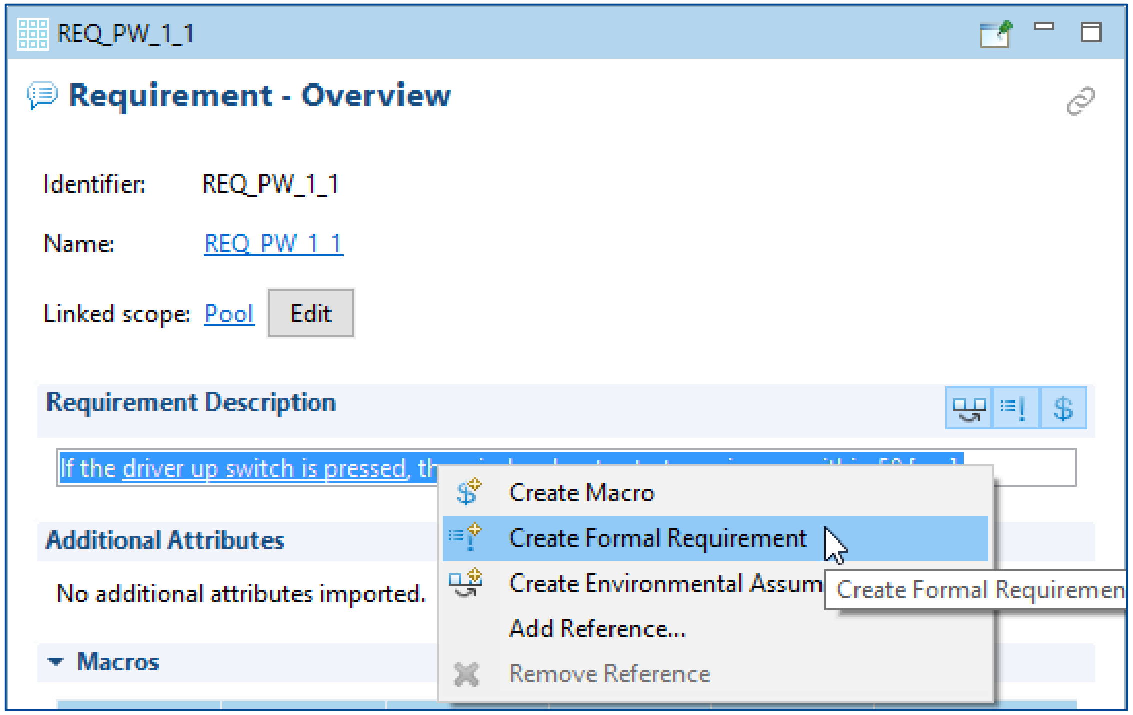
Hint: Additional information about Macros, Patterns and Formal Requirements will be given on slide "Additional Information II"

Transformation: informal → semi-formal (2/6)

To use these Macros you need to create a Formal Requirement.

1. Select the complete requirement text
2. Right-click on the text and select "Create Formal Requirement"

This will create a formal requirement that is associated with the original requirement.



REQ_PW_1_1

Requirement - Overview

Identifier: REQ_PW_1_1

Name: REQ_PW_1_1

Linked scope: Pool Edit

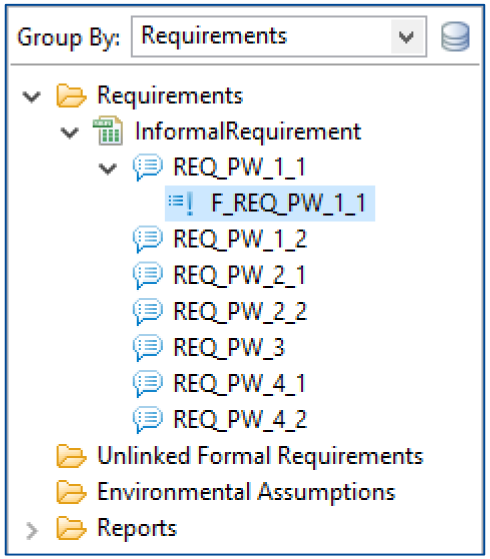
Requirement Description

If the driver up switch is pressed, th

Additional Attributes

No additional attributes imported.

Macros



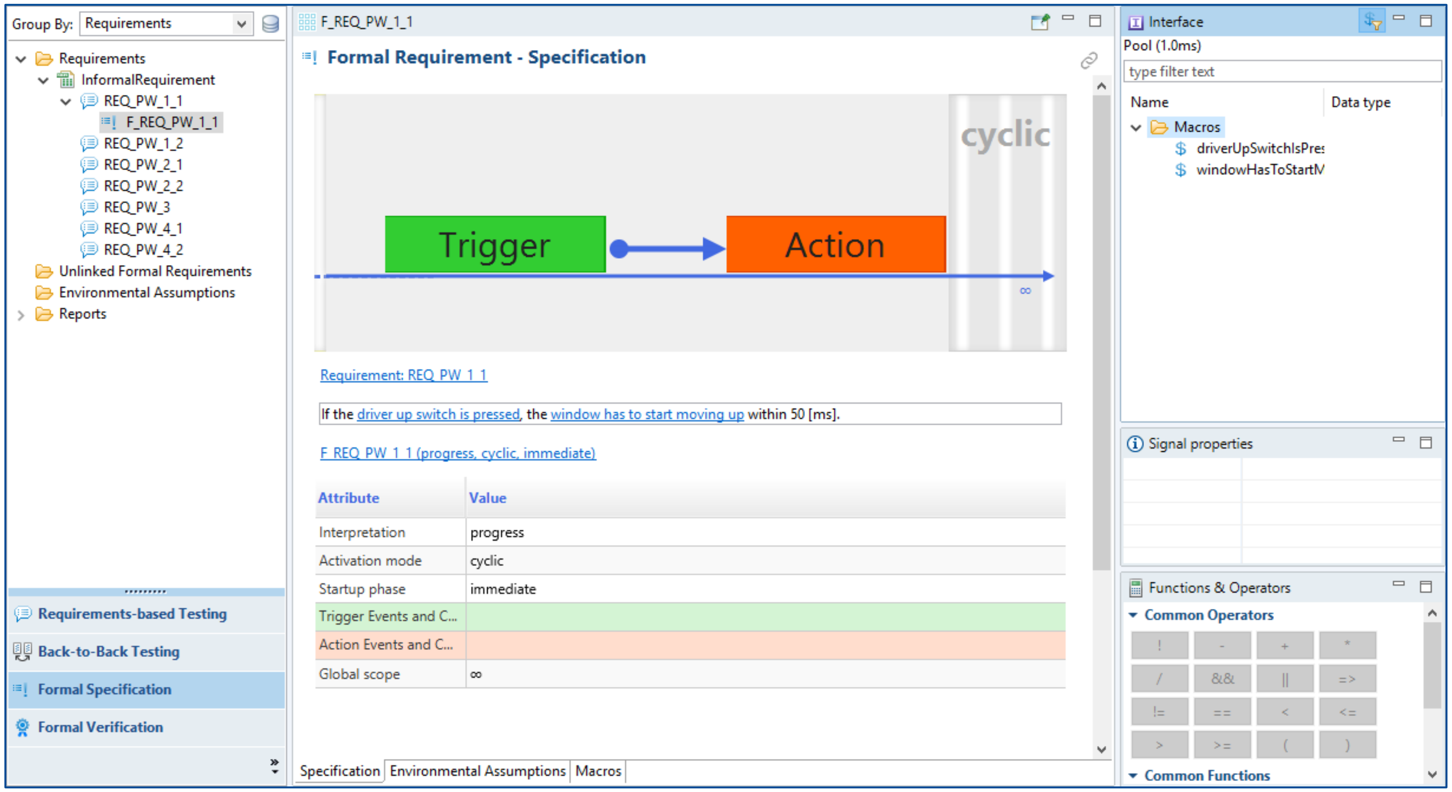
Group By: Requirements

Transformation: informal → semi-formal (3/6)

In the next step we will define the structure and the syntax of the Formal Requirement which means the relationship between the macros needs to be defined.

- Select the Formal Requirement “F_REQ_PW_1_1” in the Profile Navigator

The dashboard area will change accordingly and show the “Simplified Universal Pattern Editor”.



Requirement: REQ_PW_1_1

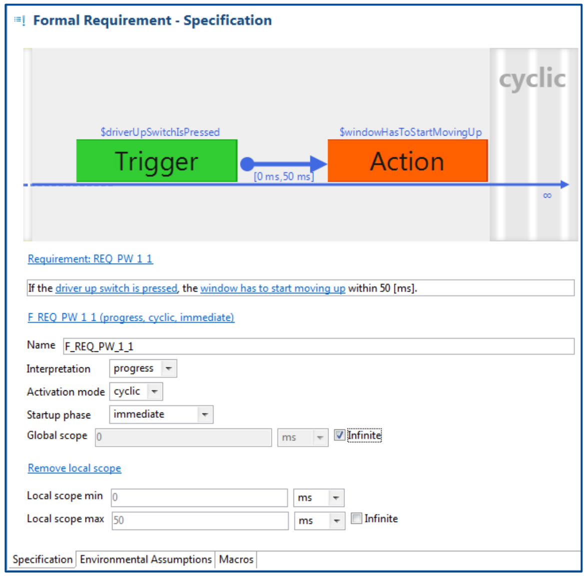
If the driver up switch is pressed, the window has to start moving up within 50 [ms].

F_REQ_PW_1_1 (progress, cyclic, immediate)

Attribute	Value
Interpretation	progress
Activation mode	cyclic
Startup phase	immediate
Trigger Events and C...	
Action Events and C...	
Global scope	∞

Hint: Additional information about Simplified Universal Patterns and their parameters will be given on the following slide.

Additional Information I



Formal Requirement - Specification

Requirement: REQ_PW_1_1

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

F_REQ_PW_1_1 (progress, cyclic, immediate)

Name	F_REQ_PW_1_1		
Interpretation	progress		
Activation mode	cyclic		
Startup phase	immediate		
Global scope	0	ms	☑ Infinite

Remove local scope

Local scope min	0	ms	
Local scope max	50	ms	☐ Infinite

Specification | Environmental Assumptions | Macros

To get a better understanding of the concept of Simplified Universal Pattern, please have a look at the diagram which is visible in the middle of the Specification board. This diagram describes a trigger condition “Trigger” and an action condition “Action”. If you have a look on the pattern parameters which will show up underneath the graph you can see that the “Interpretation” is set to “Progress”. This means that the execution waits for the trigger and then the action condition has to occur in within a given time (“Duration”); in this case 50 ms.

What do the different interpretation types mean?

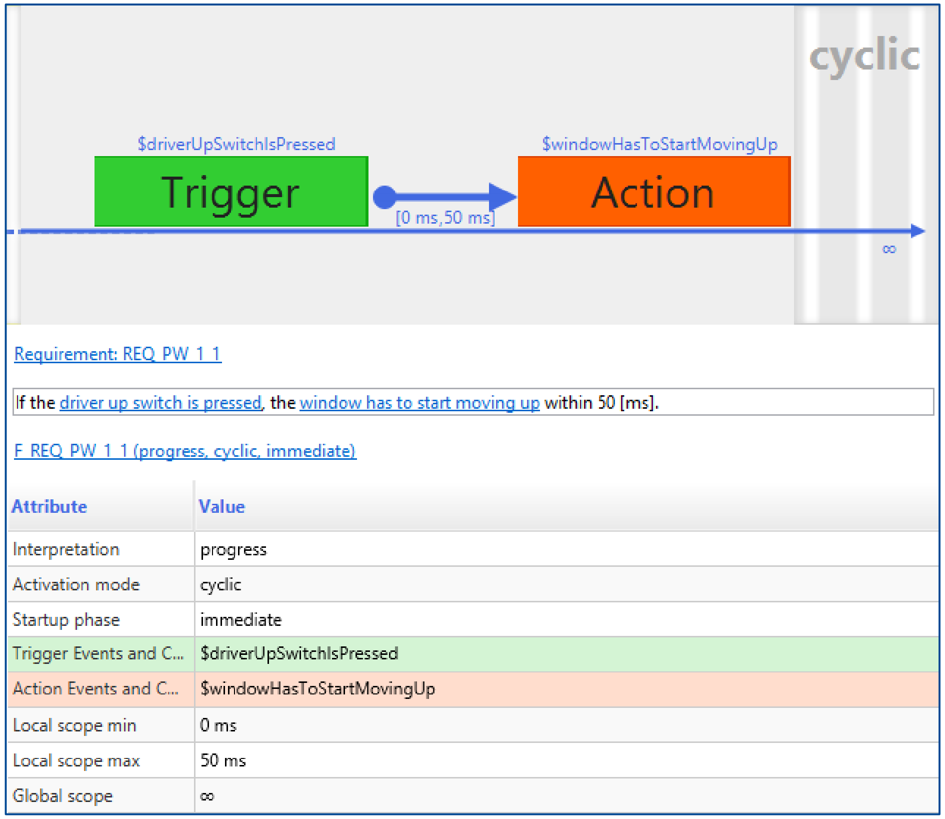
Progress: This means that if a trigger condition is fulfilled, the following action must be fulfilled in the given time too, to match the requirement specification. Otherwise the requirement is violated.

Invariant: This means that there is no trigger or to be more specific the trigger is in parallel to the action. Only the action will be observed.

Ordering: This is the opposition of “Progress”. This means that if the action is observed it will be checked if the trigger has to occur before. The specification is only fulfilled, if both events could be observed as specified.

Transformation: informal → semi-formal (4/6)

1. Click on the “Trigger” icon inside the top area of the dashboard.
2. Click inside the “Events and Condition” input field once and then double click on the Macro “driverUpSwitchIsPressed” which can be found in the Interface View inside the folder “Macros” (on the right side of the GUI).
3. Click on the “Action” icon inside the top area of the dashboard.
4. Repeat step 2 → This time use the Macro “windowHasToStartMovingUp”.
5. Click on the arrow in between the “Trigger” and the “Action” button.
6. Add a local scope by pressing the respective link.
7. In the Local scope max input field, enter ‘50’ as this is the maximum amount of time that is accepted between the trigger and the action.



Requirement: REQ_PW_1_1

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

F_REQ_PW_1_1 (progress, cyclic, immediate)

Attribute	Value
Interpretation	progress
Activation mode	cyclic
Startup phase	immediate
Trigger Events and C...	$driverUpSwitchIsPressed
Action Events and C...	$windowHasToStartMovingUp
Local scope min	0 ms
Local scope max	50 ms
Global scope	∞

Additional Information II

Information about Macros, Simplified Universal Patterns and Formal Requirements

The goal of identifying **Macros** is to enable the transformation of an informal requirement specification given in a natural language into a semi-formal specification format by identifying relevant objects in the given text.

These macros are used as placeholders to identify all kinds of events, conditions and timing information of the requirement in order to make them useable in further structuring process steps. Therefore they are visible in the dedicated “Macro” tab of the Dashboard and in the Interface View. Since the macros are directly related to the original requirement, the unambiguousness and traceability are given at any time.

Simplified Universal Patterns are used to structure the textual requirement parts (macros) and to bring them into a logical context in order to achieve a machine-readable character. BTC EmbeddedPlatform provides a flexible mechanism that covers most use cases in a safety critical requirement specification.

Formal Requirements represent the specified requirement as the result of the formalization process. They can be interpreted as kind of a container which contains the semi-formal resp. the formal representation of one or more requirements.

Transformation: informal → semi-formal (5/6)

Please repeat the performed steps from slides 11 to 14 with the following requirement Req_PW_4_1.

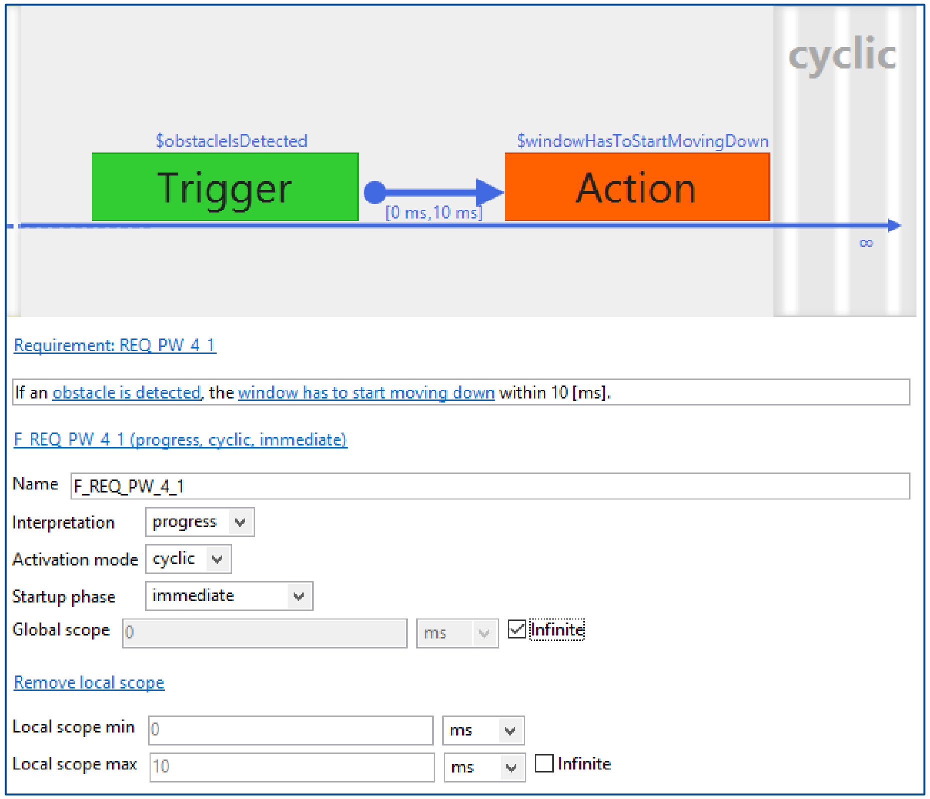
If an obstacle is detected, the window has to start moving down within 10 [ms].

Please define the following macros from the given requirement.

1. anObstacleIsDetected
2. windowHasToStartMovingDown → The tool will ask you, if you want to create a new macro or if you want to reuse the macro *$windowHasToStartMovingUp*. This macro is indeed similar but expresses the opposite, so you need to create a new macro. Please press “Create Macro”. This always happens, when the text is quite similar and is useful to avoid doubled macros with the same meaning.
3. Mark the complete text and select “Create Formal Requirement” from the context menu.

Transformation: informal → semi-formal (6/6)

1. Select the Macro "anObstacleIsDetected" as the trigger condition.
2. Select the Macro "windowHasToStartMovingDown " as the action condition.
3. Add '10 ms' as the local scope max for this requirement. The local scope is represented by the arrow in the graphical editor.



Requirement: REQ_PW_4_1

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

F_REQ_PW_4_1 (progress, cyclic, immediate)

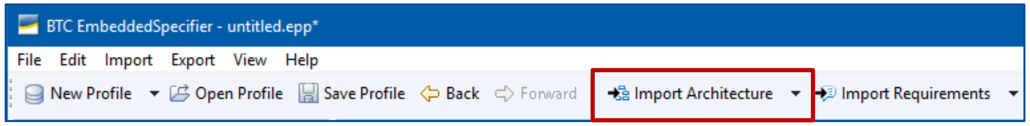
Name	F_REQ_PW_4_1		
Interpretation	progress		
Activation mode	cyclic		
Startup phase	immediate		
Global scope	0	ms	☑ Infinite

Remove local scope

Local scope min	0	ms	
Local scope max	10	ms	☐ Infinite

Import Architecture

To transform the semi-formal into a Formal Specification it is necessary to import an existing system architecture and map the given macros to the available interface objects.

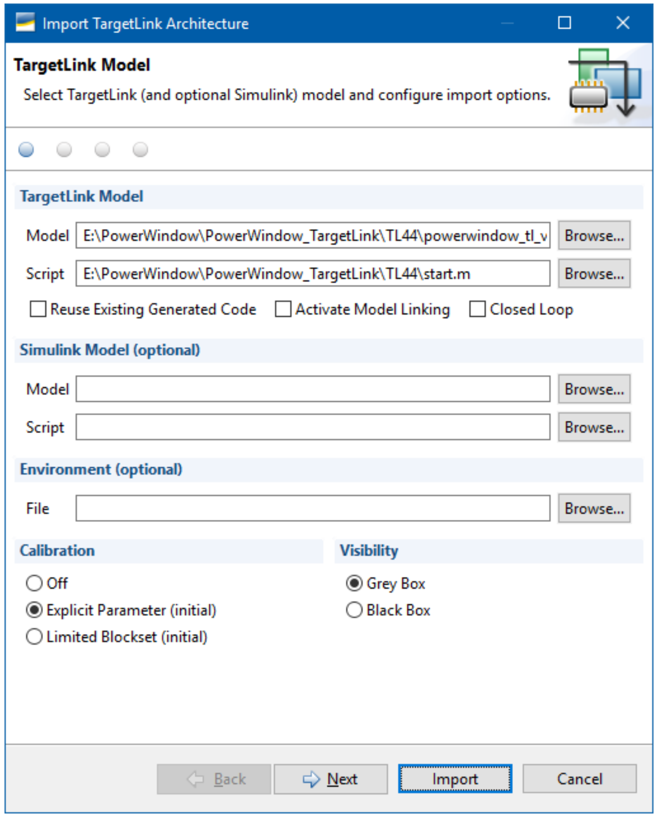


To continue click on the "Import Architecture" icon on the top toolbar. This will open up a new dialog where you can specify the data to import the TargetLink architecture. **Hint**: The described workflow can be applied for other architectures in the same manner.

In this dialog you can specify the TargetLink model and a startup script (.m-file) if necessary.

Please select the powerwindow_tl_v03.mdl file from the tutorial folder taking your current TargetLink version into account and select in addition the start.m script as well.

Please press the "Import" button to continue.


Import TargetLink Architecture

TargetLink Model

Select TargetLink (and optional Simulink) model and configure import options.

TargetLink Model

Model: E:\PowerWindow\PowerWindow_TargetLink\TL44\powerwindow_tl_v

Script: E:\PowerWindow\PowerWindow_TargetLink\TL44\start.m

☐ Reuse Existing Generated Code ☐ Activate Model Linking ☐ Closed Loop

Simulink Model (optional)

Model:

Script:

Environment (optional)

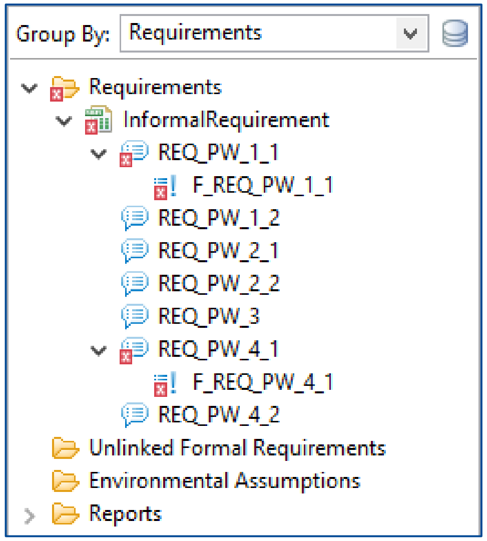
File:

Calibration: ○ Off [✓ Explicit Parameter (initial)] ○ Limited Blockset (initial)

Visibility: [✓ Grey Box] ○ Black Box

Hint: Additional information about the different import options can be found in the Profile Creation tutorial which can be found inside the documentations folder.

Transformation semi-formal → formal (1/2)

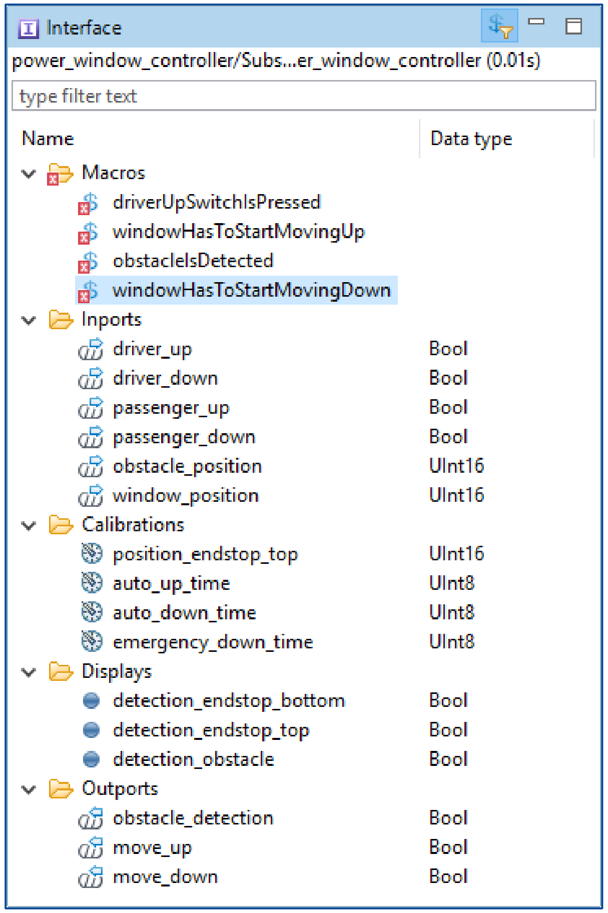


Group By: Requirements

- Requirements
  - InformalRequirement
    - REQ_PW_1_1
      - F_REQ_PW_1_1
    - REQ_PW_1_2
    - REQ_PW_2_1
    - REQ_PW_2_2
    - REQ_PW_3
    - REQ_PW_4_1
      - F_REQ_PW_4_1
    - REQ_PW_4_2
- Unlinked Formal Requirements
- Environmental Assumptions
- Reports

After the import of the architecture, please group by requirements and unfold the structure of the requirements.

As displayed in the screenshot on the left, the formal requirements are highlighted by a red icon. This means that the tool requires more information for these elements and the (formal) specification is not yet complete.



Interface

power_window_controller/Subs...er_window_controller (0.01s)

type filter text

Name	Data type
Macros	
driverUpSwitchIsPressed	
windowHasToStartMovingUp	
obstacleIsDetected	
windowHasToStartMovingDown	
Inports	
driver_up	Bool
driver_down	Bool
passenger_up	Bool
passenger_down	Bool
obstacle_position	UInt16
window_position	UInt16
Calibrations	
position_endstop_top	UInt16
auto_up_time	UInt8
auto_down_time	UInt8
emergency_down_time	UInt8
Displays	
detection_endstop_bottom	Bool
detection_endstop_top	Bool
detection_obstacle	Bool
Outports	
obstacle_detection	Bool
move_up	Bool
move_down	Bool

In addition to that, the Interface View (as shown in the screenshot on the right) now contains the architecture objects that are available inside the TargetLink model.

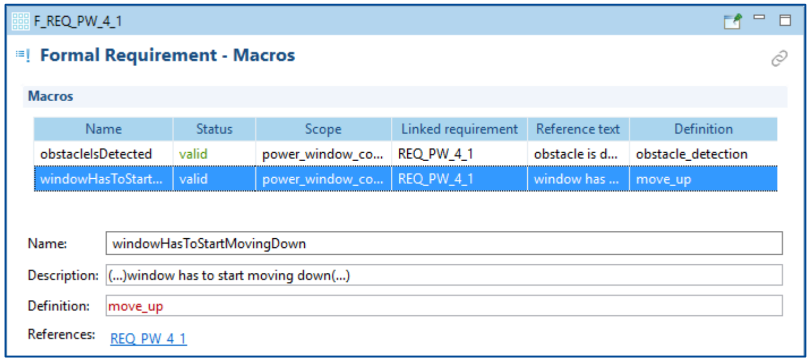
This information will be used to define the exact meaning for each macro, the step from a semi-formal specification to a formal, machine-readable specification.

Transformation semi-formal → formal (2/2)

The imported interface objects will be used to define the meaning of the macros.

1. Click on the “Macros” tab at the bottom of the main window
2. Select the macro “anObstacleIsDetected” in the table
3. Click inside the “Definition” text field below the table.
4. Double click on the icon of the outport “obstacle_detection” in the Interface View
5. Repeat steps 1-3 for all the macros
   - windowHasToStartMovingDown → move_down (OutPort)
   - windowHasToStartMovingUp → move_up (OutPort)
   - driverUpSwitchIsPressed → driver_up (InPort)

The red marks in the Specification tree automatically disappear when the definition is complete.



F_REQ_PW_4_1

Formal Requirement - Macros

Macros

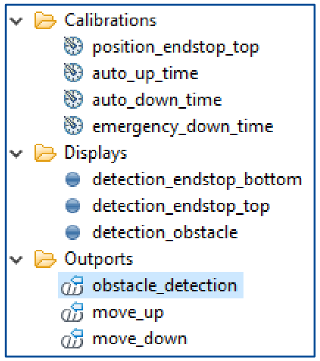
Name	Status	Scope	Linked requirement	Reference text	Definition
obstacleIsDetected	valid	power_window_co...	REQ_PW_4_1	obstacle is d...	obstacle_detection
windowHasToStart...	valid	power_window_co...	REQ_PW_4_1	window has ...	move_up

Name: windowHasToStartMovingDown

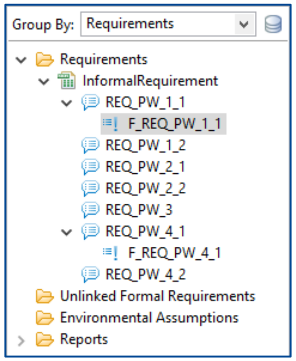
Description: (...)window has to start moving down(...)

Definition: move_up

References: REQ PW 4 1



- Calibrations
  - position_endstop_top
  - auto_up_time
  - auto_down_time
  - emergency_down_time
- Displays
  - detection_endstop_bottom
  - detection_endstop_top
  - detection_obstacle
- Outports
  - obstacle_detection
  - move_up
  - move_down



Group By: Requirements

- Requirements
  - InformalRequirement
    - REQ_PW_1_1
      - F_REQ_PW_1_1
    - REQ_PW_1_2
    - REQ_PW_2_1
    - REQ_PW_2_2
    - REQ_PW_3
    - REQ_PW_4_1
      - F_REQ_PW_4_1
    - REQ_PW_4_2
- Unlinked Formal Requirements
- Environmental Assumptions
- Reports

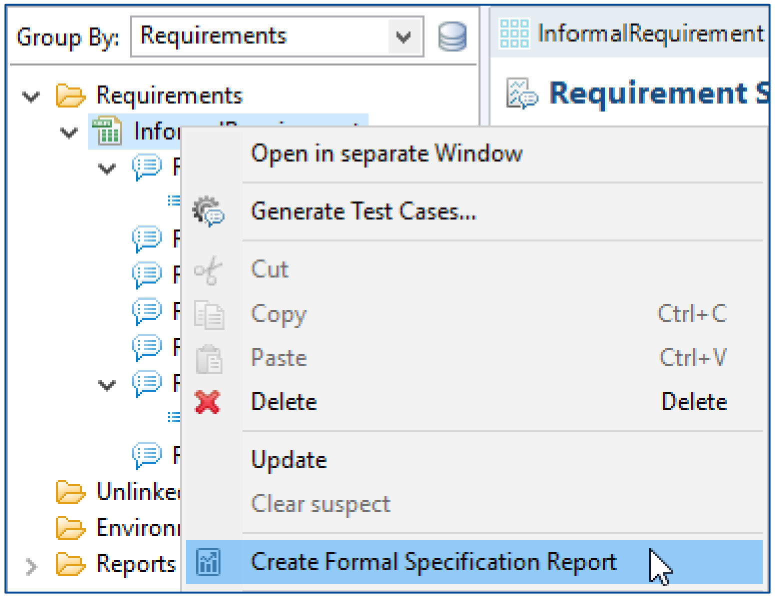
That’s it. The informal requirements have been transformed into formal requirements by establishing a direct connection between the meaningful text parts (macros) and the TargetLink systems architecture.

BTC embedded systems

Reporting (1/2)

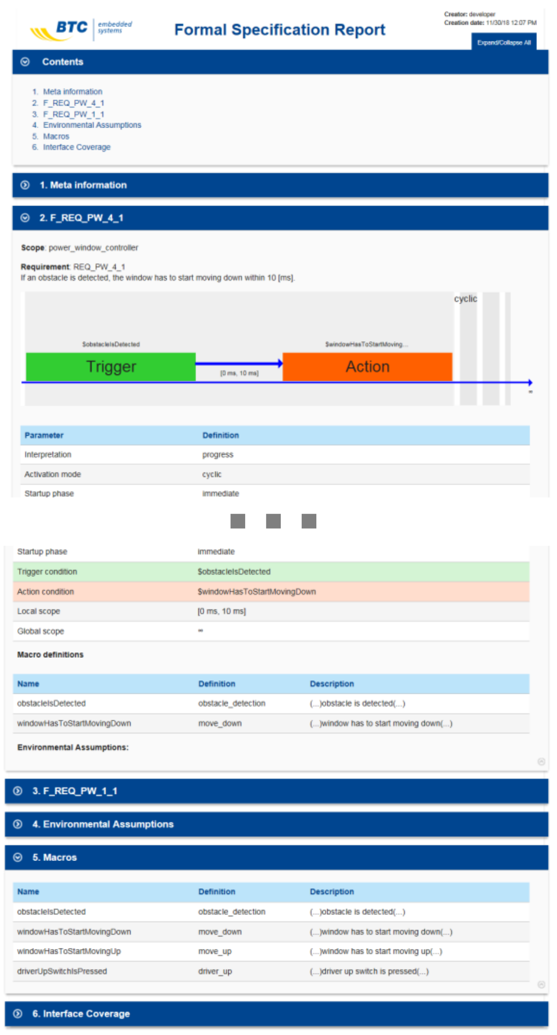
During testing, the dashboards show the current status of the test project, related to the selected item in the Profile Navigator. In addition it is sometimes necessary to export these information for documentation of the current status or presenting the test status to the customer. For the Formal Specification BTC EmbeddedPlatform provides the Formal Specification report.

To create the report, right-click on the requirements source and select Create Formal Specification Report from the upcoming context menu.



Reporting (2/2)

Formal Specification Report



Creator: developer
Creation date: 11/30/18 12:07 PM

Expand/Collapse All

Contents

1. Meta information

2. F_REQ_PW_4_1

Scope: power_window_controller

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

Parameter	Definition
Interpretation	progress
Activation mode	cyclic
Startup phase	immediate
Trigger condition	$obstacleIsDetected
Action condition	$windowHasToStartMovingDown
Local scope	[0 ms, 10 ms]
Global scope	∞

Macro definitions

Name	Definition	Description
obstacleIsDetected	obstacle_detection	(...)obstacle is detected(...)
windowHasToStartMovingDown	move_down	(...)window has to start moving down(...)

Environmental Assumptions:

3. F_REQ_PW_1_1

4. Environmental Assumptions

5. Macros

Name	Definition	Description
obstacleIsDetected	obstacle_detection	(...)obstacle is detected(...)
windowHasToStartMovingDown	move_down	(...)window has to start moving down(...)
windowHasToStartMovingUp	move_up	(...)window has to start moving up(...)
driverUpSwitchIsPressed	driver_up	(...)driver up switch is pressed(...)

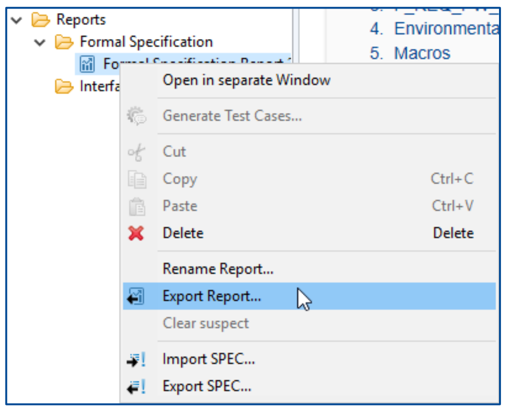
6. Interface Coverage

The Formal Specification Report is showing the textual and formalized requirements as well as Assumptions and the used macros. Each section can be expanded or collapsed.

The available section are:

1. Meta information
2. Requirements (textual and formal)
3. Environmental Assumptions
4. Macros
5. Interface Coverage

The report can easily be exported with a right-click on the report in the Profile Navigator and select "Export Report …" from the context menu.



Next Steps

The formalized requirements can be used to dramatically improve the efficiency and quality of the test process. Please refer to the following tutorials to find out more about the different use cases.

- Tutorial for Formal Test and Requirements-based test generation.pdf
- Tutorial for Formal Verification.pdf
- Tutorial for Real Time Testing Observer.pdf

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