

Automation Sequence Builder V4.2.3

User Guide



Copyright

The data in this document may not be altered or amended without special notification from ETAS GmbH. ETAS GmbH undertakes no further obligation in relation to this document. The software described in it can only be used if the customer is in possession of a general license agreement or single license. Using and copying is only allowed in concurrence with the specifications stipulated in the contract.

Under no circumstances may any part of this document be copied, reproduced, transmitted, stored in a retrieval system or translated into another language without the express written permission of ETAS GmbH.

© Copyright 2015 ETAS GmbH, Stuttgart

The names and designations used in this document are trademarks or brands belonging to the respective owners.

Document Automation Sequence Builder User Guide V4.2.3

Contents

1	Introduction.....	8
1.1	Definitions and Abbreviations.....	8
1.2	Conventions.....	8
2	Automation Sequence Builder	9
2.1	Features of the Automation Sequence Builder	9
2.2	Automation Sequence Builder GUI.....	10
2.3	Debugging.....	25
2.4	Hints for working with the ASB.....	27
2.4.1	Drag & Drop (Move vs. Copy)	27
2.4.2	Drag & Drop 'highlighting'	27
2.4.3	The help and the result list	27
2.4.4	Error indications.....	29
2.4.5	SuTM Mapping File	29
2.4.6	Renaming Stencils.....	29
2.4.7	Renaming variables.....	29
2.4.8	Generation of code out of the sequence	29
2.4.9	Creation of different kinds of projects.....	29
2.4.10	Offline testing and Preparation of an offline test bench.....	30
2.4.11	Stencil Compatibility	31
2.4.12	Hotkeys	31
2.4.13	Stencil Collapse/Expand All Hotkey (Ctrl + Space)	33
3	Stencil Description.....	34
3.1	Common.....	34
3.1.1	Define Variable	34
3.1.2	Use Variable	34
3.1.3	C #	36
3.1.4	Arithmetic.....	36
3.1.5	Constant.....	38
3.1.6	Global	38
3.1.7	Result.....	39
3.1.8	Nested Sequence	40
3.1.9	FunctionUsage	42
3.1.10	Sleep.....	44
3.2	Curve Evaluation	44
3.2.1	Datalogger	44
3.2.2	Generate Plot.....	45
3.2.3	CurveValueComparison	47
3.2.4	Define Curve Set.....	52
3.2.5	Use Curve Set.....	53
3.3	Diagnostic.....	54
3.3.1	Define Diagnostic.....	54
3.3.2	Use Diagnostic.....	55
3.3.3	Diagnostic Diag.....	55
3.3.4	DiagnosticEvaluation	56

3.4	Documentation	57
3.4.1	Report Text	57
3.4.2	Report Value.....	59
3.4.3	Comment	60
3.5	Signal Generator:	60
3.5.1	Configure SG	60
3.5.2	Action SG:	62
3.6	Fault Simulation	62
3.6.1	Fault Simulation FS	62
3.7	Model.....	65
3.7.1	Set Model Value	65
3.7.2	Get Model Value.....	66
3.8	ECU.....	67
3.8.1	Set ECU Value.....	67
3.8.2	Get ECU Value	68
3.8.3	Get ECU Mea Value	69
3.9	Structures.....	70
3.9.1	Group.....	70
3.9.2	Loop	71
3.9.3	WhileDo	72
3.9.4	DoUntil.....	73
3.9.5	Length	74
3.10	Verification	75
3.10.1	Condition	75
3.10.2	Logic Operation.....	76
3.10.3	Pass	77
3.10.4	Fail.....	78
3.10.5	If.....	78
3.10.6	Verify	80
4	Use AUTOMATION SEQUENCE BUILDER with ODX-LINK 1.4	82
4.1	Prerequisites.....	82
4.2	Test bench configuration	82
4.3	Tool configuration	82
4.4	ODX services configuration	83
4.5	Test case.....	86
5	ETAS Contact Addresses	87

Figures

Figure 1: Favorites	9
Figure 2: ASB Welcome Screen	11
Figure 3: Sequence Editor	11
Figure 4: ASB User Interface.....	12
Figure 5: Toolbar	14
Figure 6: Assembling a Sequence.....	15
Figure 7: Including "Get Model Value" in the "Condition" stencil.....	16
Figure 8: Engine check.....	16
Figure 9: Engine check result.....	17
Figure 10: Change display size.....	17
Figure 11: Save sequence	18
Figure 12: Data recording.....	18
Figure 13: "Select Generators" window.....	20
Figure 14: "ATCLGenerator" window.....	21
Figure 15: "Messages" Tab.....	22
Figure 16: "ASBGenerator" window.....	22
Figure 17: Test Project directory.....	22
Figure 18: ASQ Batch Generator	23
Figure 19: Toggle Breakpoint	24
Figure 20: "Information Panel"	24
Figure 21: Toggle Breakpoint	26
Figure 22: "Debug Messages" tab.....	26
Figure 23: Toolbar	26
Figure 24: Separation line	27
Figure 25: Resizing Columns.....	27
Figure 26: 'Hand' icon on top of the "Value" field.....	31
Figure 27: Stencils having 'Hand' icon.....	31
Figure 28: Change system hotkeys for changing the language.....	33
Figure 29: "Define Variable" Stencil.....	34
Figure 30: "Use Variable" Stencil.....	34
Figure 31: "Use Variable" Stencil.....	35
Figure 32: "Use Variable" Stencil (Scalar).....	35
Figure 33: "C# Code" Stencil.....	36
Figure 34: "Arithmetic" Stencil.....	36
Figure 35: "Arithmetic" Stencil Example	37
Figure 36: "Constant" Stencil	38

Figure 37: "Global" Stencil..... 38

Figure 38: "Result" Stencil 39

Figure 39: "Nested Sequence" Stencil..... 40

Figure 40: Variables defined in 'getsum.asq' (Child sequence) 40

Figure 41: "Nested Sequence" Stencil Example (Parent sequence)..... 41

Figure 42: "Options" Dialog Box..... 41

Figure 43: "Select File" Dialog Box 42

Figure 44: "FunctionUsage" Stencil 43

Figure 45: "Sleep" Stencil 44

Figure 46: "Datalogger" Stencil 44

Figure 47: "Datalogger" Stencil Example 45

Figure 48: "Generate Plot" Stencil 45

Figure 49: "Generate Plot" Stencil Example..... 46

Figure 50: "CurveValueComparision" Stencil 47

Figure 51: "CurveValueComparision" Stencil 48

Figure 52: "CurveValueComparision" Stencil Example 51

Figure 53: "Define Curve Set" Stencil..... 52

Figure 54: "Define Curve Set" Stencil Example 52

Figure 55: "CurveValueComparision" Stencil..... 53

Figure 56: "Use Curve Set" Stencil 53

Figure 57: "Define Diagnostic" Stencil 54

Figure 58: "Define Diagnostic" Stencil Example 54

Figure 59: "Use Diagnostic" Stencil 55

Figure 60: "Use Diagnostic" Stencil Example 55

Figure 61: "Diagnostic Diag" Stencil..... 55

Figure 62:"Diagnostic Diag" Stencil Example 56

Figure 63: "DiagnosticEvaluation" Stencil..... 56

Figure 64: "DiagnosticEvaluation" Stencil Example 57

Figure 65: "Report Text" Stencil 57

Figure 66: "Report Value" Stencil..... 59

Figure 67: "Report Value" Example..... 59

Figure 68: "Report Value" Stencil Output in a Report 60

Figure 69: "Comment" Stencil 60

Figure 70: "Configure SG" Stencil..... 61

Figure 71: "Action SG" Stencil 62

Figure 72: "Fault Simulation FS" Stencil 62

Figure 73: "Fault Simulation FS" Stencil Example 64

Figure 74: "Set Model Value" Stencil 65

Figure 75: "Set Model Value" Stencil Example..... 65

Figure 76: "Get Model Value" Stencil..... 66

Figure 77:"Get Model Value" Stencil Example..... 66

Figure 78: "Set ECU Value" Stencil 67

Figure 79: "Set ECU Value" Stencil Example 68

Figure 80: "Get ECU Value" Stencil 68

Figure 81:"Get ECU Value" Stencil Example..... 69

Figure 82: "Get ECU Mea Value" Stencil 69

Figure 83: "Group" Stencil..... 70

Figure 84: "Group" Stencil Example 70

Figure 85: "Loop" Stencil..... 71

Figure 86: "Loop" Stencil Example 71

Figure 87: "WhileDo" Stencil..... 72

Figure 88: "WhileDo" Stencil Example 72

Figure 89: "DoUntil" Stencil 73

Figure 90: "DoUntil" Stencil Example..... 73

Figure 91: "Length" Stencil..... 74

Figure 92: "Length" Stencil Example 74

Figure 93: "Condition" Stencil 75

Figure 94: "Condition" Stencil Example 75

Figure 95: "Logic Operation" Stencil 76

Figure 96: "Logic Operation" Stencil Example..... 76

Figure 97: "Pass" Stencil..... 77

Figure 98: "Pass" Stencil Example 77

Figure 99: "Fail" Stencil 78

Figure 100: "If" Stencil..... 78

Figure 101: "If" Stencil Example 79

Figure 102: "Verify" Stencil 80

Figure 103: "Verify" Stencil Example..... 80

Figure 104: "Verify" Stencil usage depicted by "If" stencil Example..... 81

Figure 105: Test bench configuration for ODX-Link 1.4 82

Figure 106: Tool configuration for ODX-LINK 1.4 82

Figure 107: ASB test case using diagnosis stencils 86

1 Introduction

This document provides descriptions of the Automation Sequence Builder (ASB) and helps you to

- Obtain an overview of ASB,
- Features of ASB,
- Description of stencils available in ASB
- Useful tips and hints

Whenever you need some information about usage of Automation Sequence Builder, this document should be your first entry point to find a solution for your issue.

1.1 Definitions and Abbreviations

ASB

Automation Sequence Builder

ATCL

Automotive Testing Class Library

ECU

Electronic Control Unit

SDF

Switch Definition File

SMF

SUT Mapping File

SUT

System under Test

TRL

Test Release Library

UuT

Unit under Test

1.2 Conventions

The following typographical conventions are used in this document:

<code>OCI_CANTxMessage msg0 =</code>	Code snippets are presented on a gray background and in the Courier font.
	Meaning and usage of each command are explained by means of comments. The comments are enclosed by the usual syntax for comments.
Choose File → Open .	Menu commands are shown in boldface.
Click OK .	Buttons are shown in boldface.
Press <ENTER>.	Keyboard commands are shown in angled brackets.
The "Open File" dialog box is displayed.	Names of program windows, dialog boxes, fields, etc. are shown in quotation marks.
Select the file <code>setup.exe</code>	Text in drop-down lists on the screen, program code, as well as path- and file names are shown in the Courier font.
<i>A distribution</i> is always a one-dimensional table of sample points.	General emphasis and new terms are set in italics.

2 Automation Sequence Builder

The Automation Sequence Builder (ASB) facilitates creation of test cases for users with no experience of programming software.

A test sequence is created in a simple GUI where individual parts of a sequence, in the form of blocks, are joined together and possibly parameterized. The sequences created can be saved, reopened and edited. The end of the process is the generation of C# code for test cases and the generation of test cases in the form of executables (*.exe).

Note: Representing a sequence by reading a C# code is not possible in ASB.

2.1 Features of the Automation Sequence Builder

The Automation Sequence Builder allows a very easy and handy possibility to create test cases. It does not require knowledge of the test bench architecture and how it can be addressed from the test case. Configurations of the tools are completely hidden under the shell of this tool. The focus of the tool is the test sequence itself.

Generation of test cases with graphical support

Test sequences are created by using needed actions/ structures – called stencils. Each stencil can be renamed, filled with parameters and rearranged inside the sequence. Different structuring of the sequence is possible very intuitively.

Stencils can be added to the sequence either by

1. Dragging them from the “toolbox” and dropping it
2. By using the context menu of the sequence with respect to the different value fields
3. By using the respective shortcuts defined in the respective context menus.

Note: Shortcuts will not work, if the same are used by other applications and registered at operating system level.

Favorites

Test subsequences, which you would want to use more than once, can be saved into a place from where it can be reused. The Automation Sequence Builder provides such a container called Favorites. You have to drag and drop stencils you have built, into the “Favorites” category.

Note: Only single stencils can be dragged and dropped into the “Favorites” container. Each stencil though can contain any number of stencils inside.

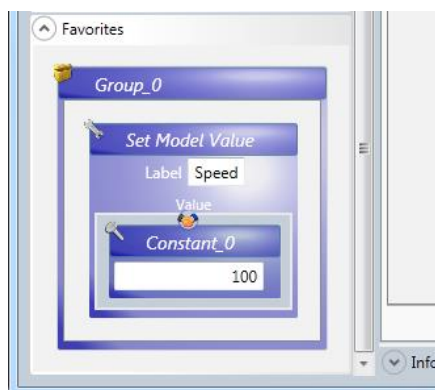


Figure 1: Favorites

Templates

Different use cases require different sequences, default inputs or guidelines. To accommodate this demand, the Automation Sequence Builder handles different templates. A template is like a grouping of specific stencils for a dedicated use case. It predefines a subset of default stencils and of your own defined favorites.

By using a template during sequence creation, only the valid stencils for the kind of sequences created are available for the developer. Also depending on the selected template, different code generators are available.

Graphical Debugging

Debugging is a very useful and powerful feature in particular for test case developers and testers. It allows you to pause the sequence execution at a dedicated step, to run through the sequence step by step and to resume the execution.

As this is a complex feature a detailed description is given in a separate chapter (see chapter 2.3 Debugging).

Reporting

The Automation Sequence Builder uses the Test Handler Commandline Application for the execution of sequences. Therefore, also the same report configuration as in Test Handler is used.

For debugging purposes the ASB writes a high amount of information regarding stepping in and out of stencils into the report. Those report entries are written with a ReportLevel of 100. (For more information about ReportLevels refer to LABCAR AUTOMATION User Guide chapter 4.6.3 Filtering Test Report Data)

The number of automatic report entries can be high, respectively the amount of information in the report and its file size increase. To avoid this, these report entries can be optionally be filtered in the Standard Report already when writing the report. Two options are in the Standard Report: "Report Level" and "Write all larger than given Report Level". The "Report Level" option sets the limit, the second option is used to define whether report entries with a ReportLevel smaller or bigger than the limit are written to the report. In either case the limit is included.

The default options are "Report Level = 0" and "Write all larger than given Report Level = true". With these options everything will be written to the report, equal to the behavior of previous versions of LABCAR AUTOMATION.

2.2 Automation Sequence Builder GUI

This section contains a short overview of the Automation Sequence Builder GUI and how to use it.

To open the Automation Sequence Builder

Select **Programs** → **ETAS** → **LABCARAUTOMATION 4.2** → **Automation Sequence Builder**

The Automation Sequence Builder opens.

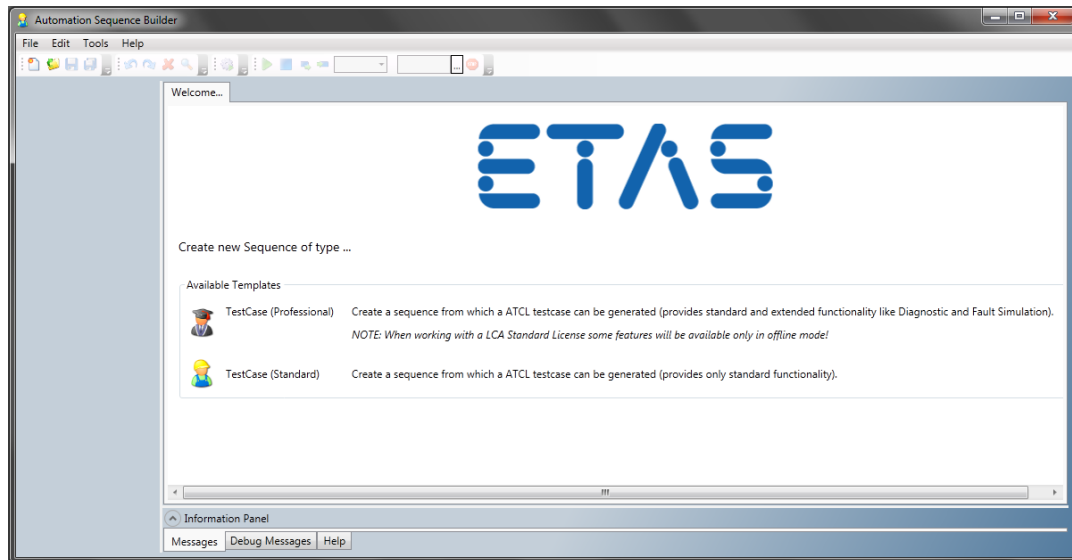


Figure 2: ASB Welcome Screen

A template can be selected from this start screen. A template describes the type of sequence that shall be created.

Different templates and their respective features are available based on the LCA license you have. The standard template and the respective features are available with the LCA standard package license. To use all features of the professional template one would need a LCA professional package license. However it is in all cases possible to create sequences of both. It would only not be possible to execute a professional sequence with a standard license if using features that are not covered by the standard license. Further templates can be purchased and added.

To select a template

Select **File** → **New** in the main menu.

Select a template from the "Templates" window and click **OK**.

A new 'Sequence Editor' tab is created. This is where you assemble your test cases.

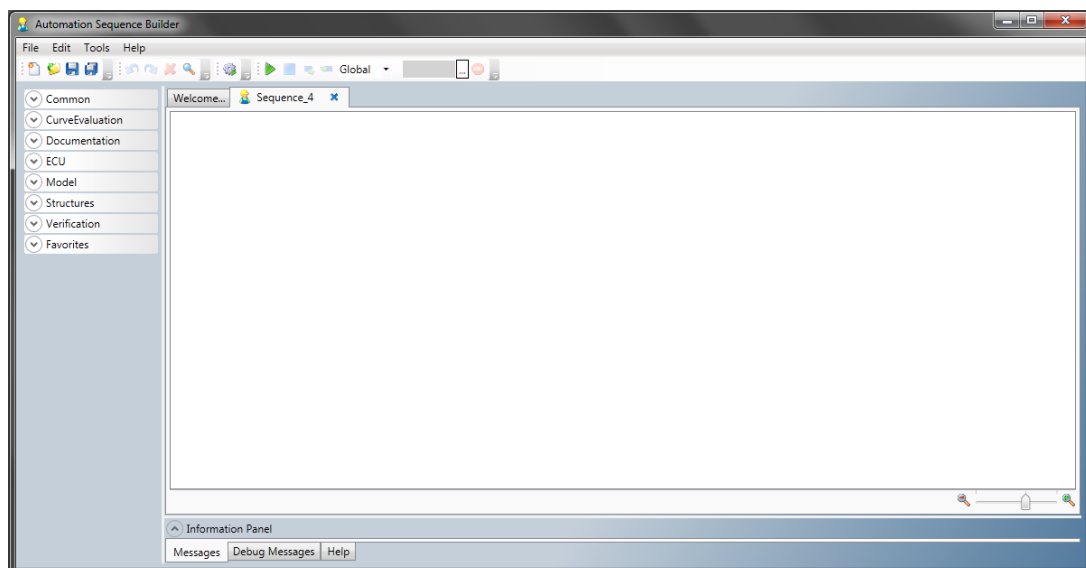


Figure 3: Sequence Editor

If you want to create test cases immediately, continue with "To assemble a sequence" on page 60. The following section is a description of the Automation Sequence Builder user interface.

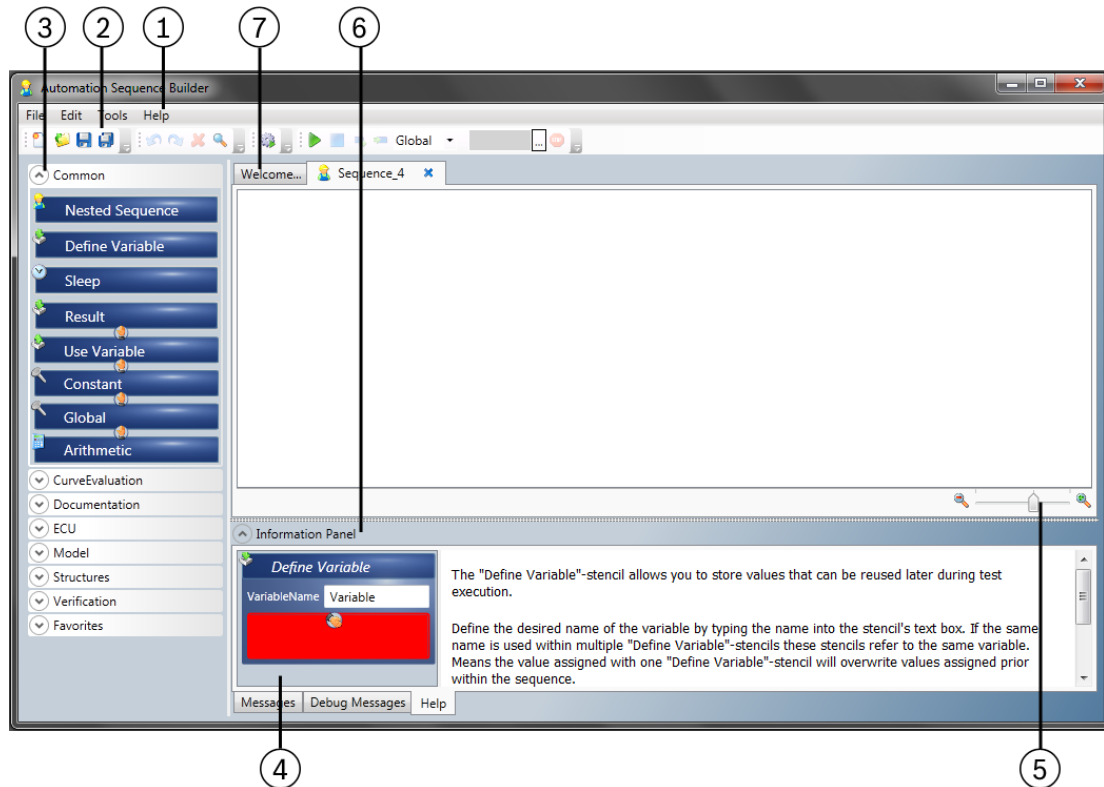


Figure 4: ASB User Interface

The user interface consists of the following elements:

1. The menu bar
2. The toolbar
3. The types of stencils
4. The information window for displaying help texts/messages
5. The magnification slider for maximizing/minimizing the display
6. The icon for hiding/showing the Information Panel
7. The tab for displaying sequences

The Menu Bar

The main menu of the Automation Sequence Builder contains the following items:

- **File**
 - **New**
To create a new sequence
 - **Open**
Opens a sequence saved previously (*.asq)
 - **Save**
Saves a sequence
 - **Save As**
Saves the sequence currently open under a different name
 - **Save All**
Saves all the open sequences
 - **Properties**
Shows information for the sequence of the tab currently selected
 - **Close Sequence**
Closes the active sequence
 - **Exit**
Closes the Automation Sequence Builder

Note: If at least one sequence was already created in the ASB and saved, a list of the last five saved sequences will additionally appear after the "Exit" menu item.

- **Edit**
 - **Undo**
Undo the previous action
 - **Redo**
Repeats an action undone
 - **Cut**
Allows you to cut a stencil to insert into another part of the sequence
 - **Copy**
Allows you to copy a stencil to insert into another part of the sequence
 - **Paste**
Allows you to paste a copied stencil into another part of the sequence
 - **Rename**
Renames the selected stencil
 - **Delete**
Deletes the selected stencil
 - **Find**
Allows you to run a search within the active sequence

- **Tools**
 - **Generate**
Starts generating the test case
 - **Plugins**
Allows you to select and activate a plugin (if available)
 - **Options**
Displays the "Options" window. In the "Options" window , you can define your test case preferences i.e., SuTMapping File, Test Bench Configuration file, Library Search Path etc.
- **Help**
 - **User's Guide**
Opens the ASB User Guide
 - **Open License Manager**
Opens the ETAS License Manager
 - **About**
Information on the Automation Sequence Builder

The Toolbar



Figure 5: Toolbar

Some of the actions in the main menu can be initiated using the icons in the toolbar.

1. **New**
See **File** → **New**
2. **Open**
See **File** → **Open**
3. **Save**
See **File** → **Save**
4. **Save All**
See **File** → **Save All**
5. **Undo**
See **Edit** → **Undo**
6. **Redo**
See **Edit** → **Redo**
7. **Delete**
See **Edit** → **Delete**
8. **Find**
See **Edit** → **Find**
9. **Generate**
See **Tools** → **Generate**
10. **Start Debugger**
Allows you to start/resume running the debugger
11. **Stop Debugger**
Allows you to stop running the debugger
12. **Step Into**
Allows you to step into a "nested stencil"

13. Step Over

Allows you to step over a "nested stencil"

14. Test Bench Selector

Allows you to select the test bench configuration where the sequence must be run. You can choose "Global", "Local" or "Offline"

15. Test Bench Configuration File Selector

Allows you to select the test bench configuration file that shall be used to run the sequence. This is applicable only for local options

16. Toggle Breakpoint

Allows you to include a breakpoint at the currently selected stencil

Sequences

At the left-hand edge of the main Automation Sequence Builder window, you will find stencils, sorted into 'Categories', which can be assembled in the form of blocks to form a test sequence.

Note: The number and type of stencils/categories vary from template to template.

Note: You can display help texts on each of the stencils in the "Information Panel" window under the "Help" tab.

Different types of stencils and their description are provided in "**Stencil Description**" section.

To assemble a sequence

- Add individual steps to a sequence by placing the relevant stencil (drag-and-drop) in the area intended for this sequence.
In the example shown, a comment was created and then a "Condition" using the "If" and "Condition" stencil. The "Condition" compares two values, taken from steps that return values. For the left-hand side of the "Condition", the "Get- Value" stencil is dragged into the relevant section; a "Constant" is used for the right-hand side in this example.

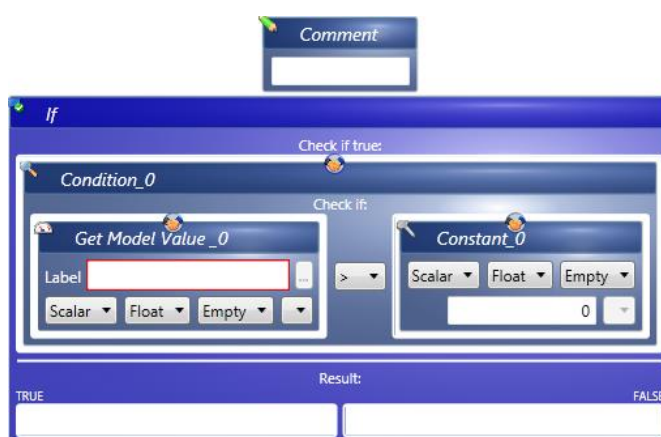


Figure 6: Assembling a Sequence

Note: When moving a stencil within a sequence by drag-and-drop, its future position is shown by a preview.

Note: If a stencil is not configured correctly yet, the corresponding fields in the stencil will be marked red.

Note: The hand beside the operands in the comparison shows that a value is required here – accordingly, only an operation stencil that returns a value (shown by the hand over the relevant icon) can be added here.

- Rename the relevant steps by double-clicking, for example, on "Get Model Value_0".

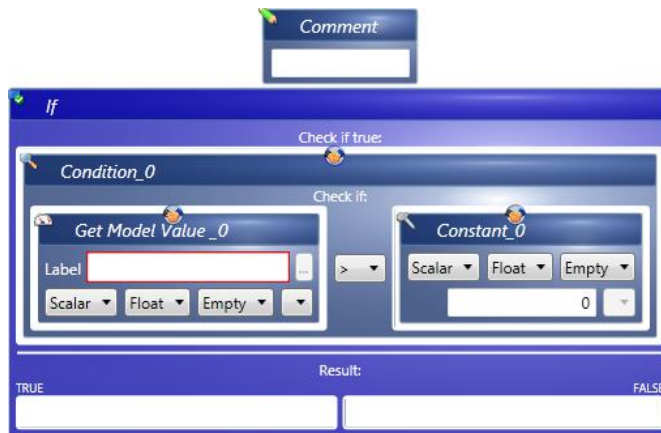


Figure 7: Including "Get Model Value" in the "Condition" stencil

- Proceed in the same way with the other steps by renaming them as shown.
 - Also enter the comment, the test label of the speed in the model and the value of the target speed.
- Or*
- Choose the label using the label picker and set the relevant speed to be checked for

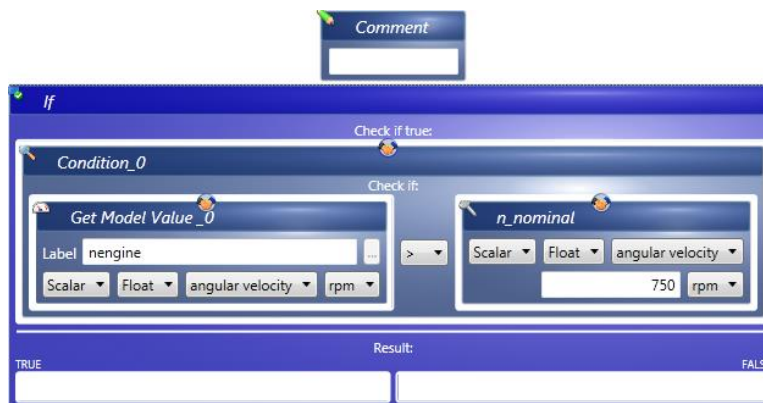


Figure 8: Engine check

- This step checks whether engine speed is greater than 750.
- Under "Result" add the relevant instruction on how to proceed in each case (condition applicable or not).

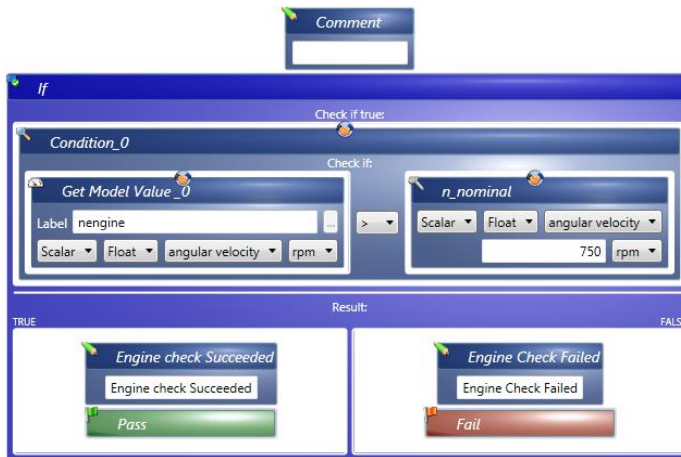


Figure 9: Engine check result

In the example, the relevant comments and verdicts are displayed.

When a test verdict within a group returns the result "fail", the entire test case receives the result "fail". A test case is considered "passed" only if all its parts receive the result "passed".

To change the display size

- To change the size of the elements of a sequence, click the magnification slider at the bottom right and move it in the appropriate direction.
Or
Scroll the mouse wheel to zoom in and out the display.

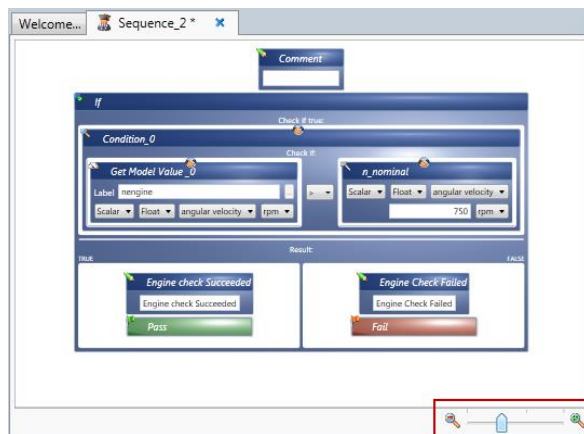


Figure 10: Change display size

Note: When using the zoom slider (or the buttons) below the sequence pane, the center of the visible part of the sequence is used as center for the zoom. When using the mouse wheel for scrolling, the position of the mouse cursor is used as center.

To save a sequence

- Select **File** → **Save As** to save a sequence for the first time. The file name of the sequence is displayed in the tab.

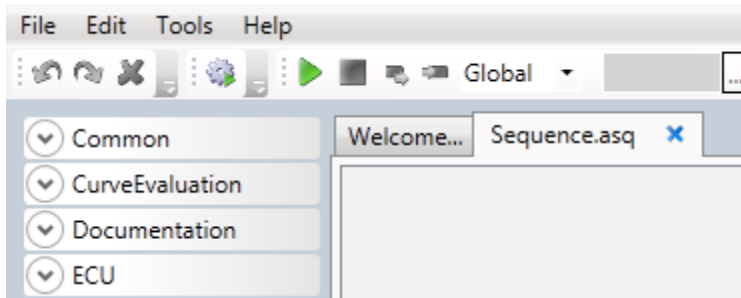


Figure 11: Save sequence

- This file is saved again with **File** → **Save**.

To add a data recording

- Drag and drop a "Datalogger" stencil into the "Define Curve Set" stencil in the sequence.
- To add a variable to the record, click on "Add Signal" in the stencil. This creates fields for "Name", "Unit" and "Acquisition Task" in the "Signals" line.
- In the "Name" field, enter the value "engine".
- In the "Unit" field, enter the value "rpm".
- Drag all previously created stencils of the sequence into the "Datalogger".
- Drag a stencil of the "Sleep" into the Datalogger stencil.
- For a recording time of 1 second, enter the value "1000".

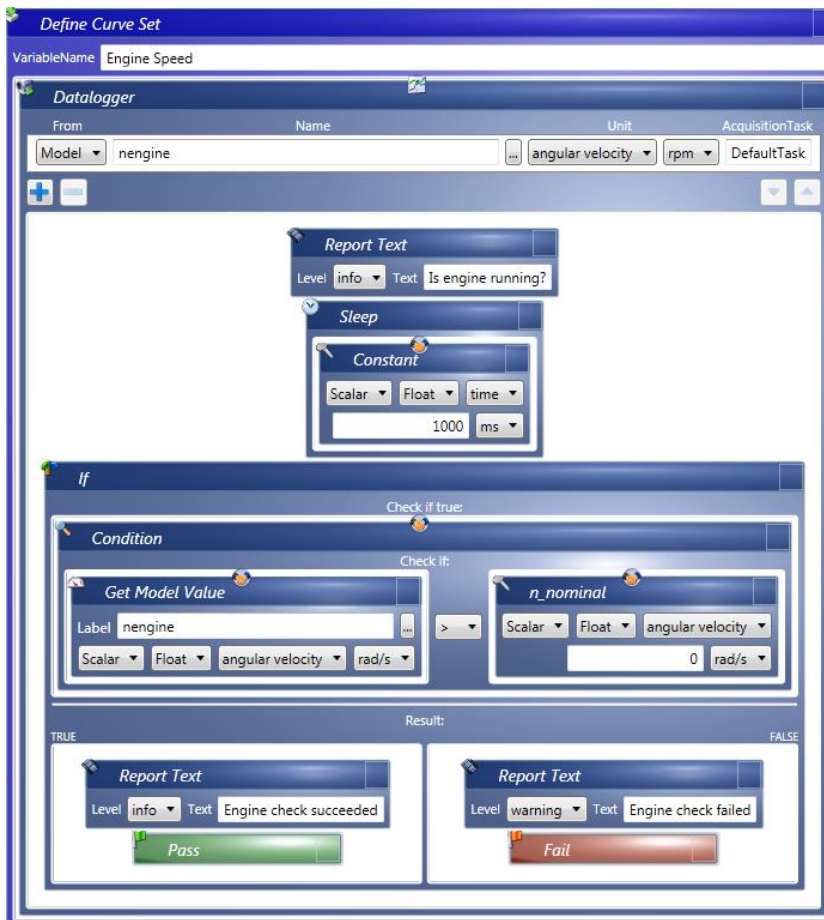


Figure 12: Data recording

In this example, recording takes place for a recording time of one second, followed by a check whether this condition is fulfilled. After completion of the functions contained in Datalogger, all recorded values are listed in a diagram and written into a report using a "Generate Plot" stencil or a "CurveValueComparison" stencil.

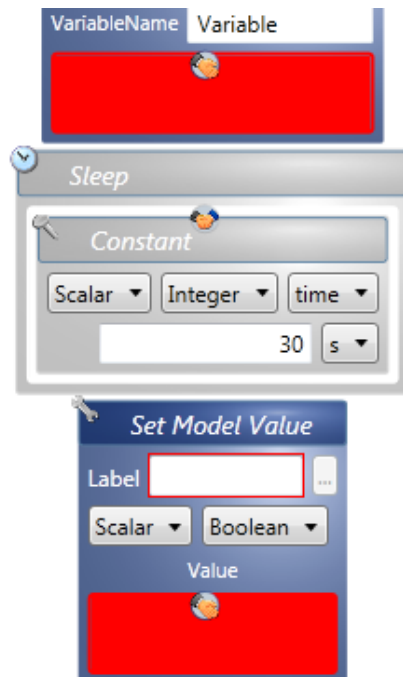
Note: The axis designations and units for the diagram are taken from the stencil. Scaling for the diagram, colors and line thicknesses are fixed automatically.

Note: The assembled reports can be viewed and evaluated in the Report Viewer.

Commenting out parts of a sequence

In order to exclude certain steps of a sequence from generation, they can be commented out.

- Select the step you want to exclude from generation.
- Right-click the step and select **Comment Out**.
- The step will be greyed out.



To generate the test case

Once you have created and saved your sequence, you can generate the test case.

- Select **Tools** → **Generate** in the main menu.
Or
- Click the **Generate** icon in the toolbar.
The "Select Generators" dialog box opens.
- Select "ATCLTestCaseASBGenerator" and click **OK**.

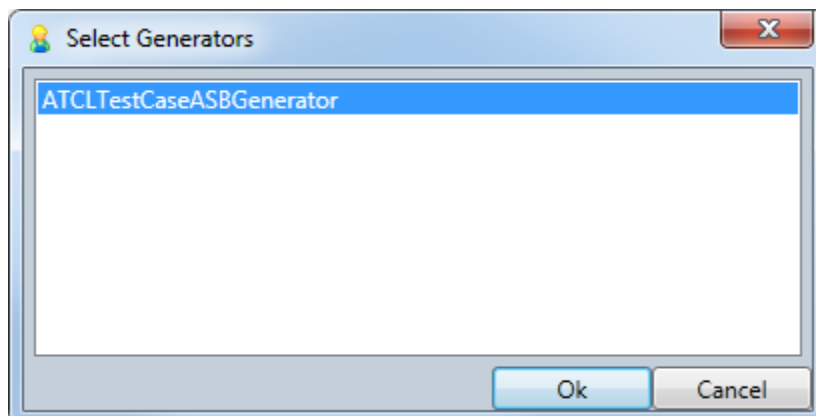


Figure 13: "Select Generators" window

- In the following window, select a directory in which you want to store the generated data or create a new one.
- Activate the desired options for code generation:
 - Offline Test Bench
Select this option if you have not installed any hardware or further tools (such as INCA) or if you want to execute the test offline for testing purposes.
 - C# Code and Executable in global Test Release Library (TRL)
Select this option if you want to generate C# code and an executable file in the Test Release Library.
 - Executable and new LABCAR-AUTOMATION Project
Select this option if you want to create an executable file and a new LABCAR-AUTOMATION Project.
 - C# Code and Visual Studio Project (2005)
Select this option if you want to generate C# code and a Visual Studio Project.
- Click **Generate**.

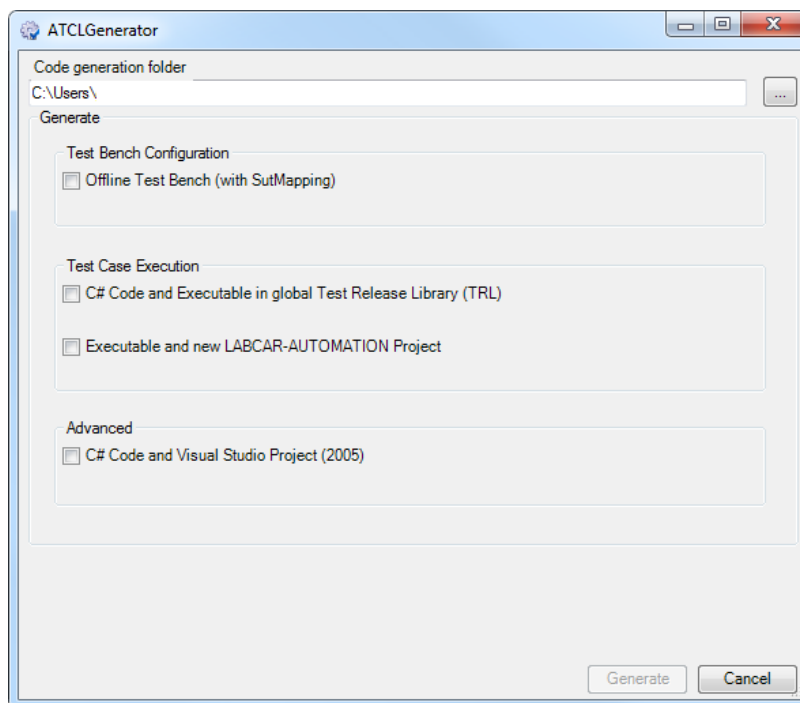


Figure 14: "ATCLGenerator" window

Note: If you have already generated a test case in the selected directory, the command **Overwrite** is available instead of the command **Generate**. In this case, files that were already created will be overwritten.

When Code generation starts, generation progress is indicated in the "Messages" tab.

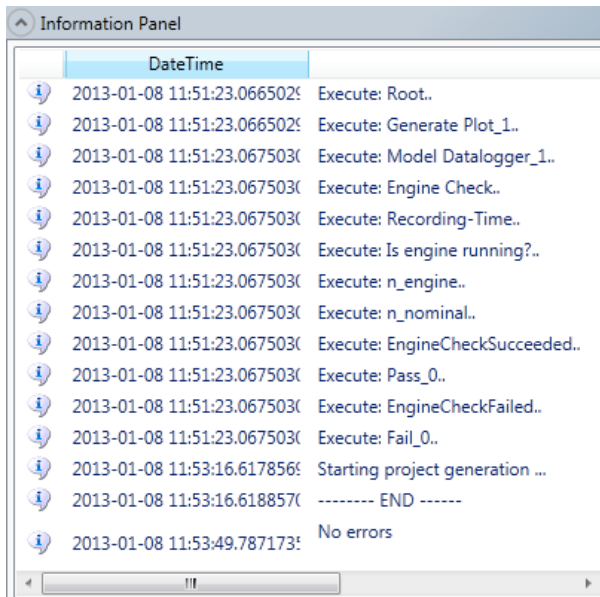


Figure 15: "Messages" Tab

ACTL-compliant C# code is generated and compiled to form an executable file. These files are stored, depending on the activate options for code generation, together with all other important files in the \TRL (Test Release Library) directory.

Successfully created files and the associated file paths are displayed in the "ASBGenerator" window.

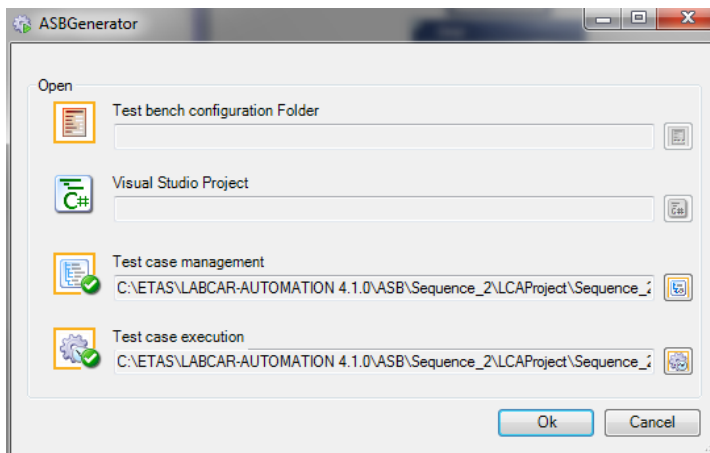


Figure 16: "ASBGenerator" window

- Click **Ok**.

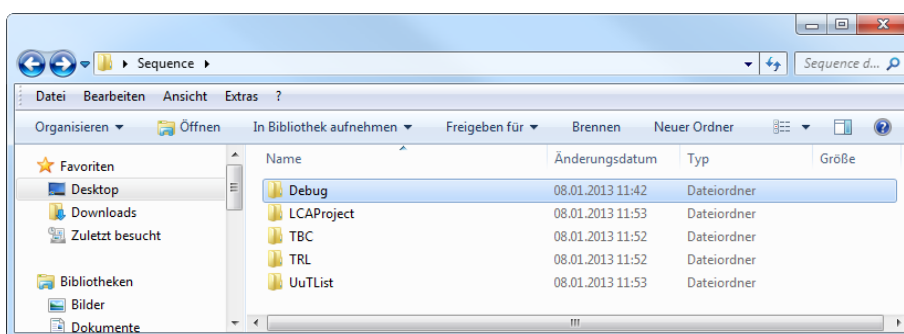


Figure 17: Test Project directory

A complete test project is also created containing this test case and all other necessary files and directories (UuTList etc.).

ASQ batch generation

The ASQ batch generator that is accessible from ASB's menu bar under **Tools** → **Plugins** → **ASQ Batch Generator** offers capability to generate LCA test cases resp. function libraries from multiple .asq files as a batch operation.

Once the generator dialog is opened 3 input parameters have to be defined:

- **Source Folder**
All *.asq file contained in the selected folder will be processed.
If the combo box "Subdirectory" is selected, even the *.asq files in subdirectories are taken into account.
- **Generator**
The generator with that th *.asq files shall be processed. Select "ATCLTestCaseASBGenerator" or "ATCLFunctionASBGenerator" depending if test executables or libraries shall be created
- **Output Folder**
The folder to that the generated test executables resp. libraries shall be stored.
Note: Files already existing in the directory will be replaced without further request!

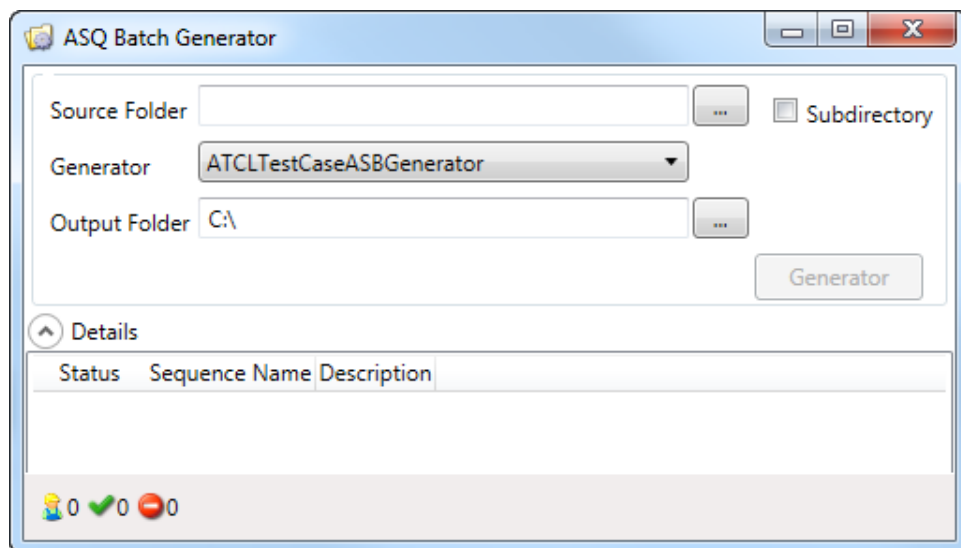


Figure 18: ASQ Batch Generator

When starting the generation by pressing the **Generator** button, the generator will collect all the *.asq files to be processed and list them in the result list. Afterwards the files will be processed one after another (what is indicated accordingly). For those files for that the generation failed a resp. error message will show up in the "Description" column of the result list. Further the status bar at the bottom will summarize the generator result (number of successfully resp. failed generations).

To start/stop debugging

- Select a test bench configuration at the top of the "Automation Sequence Builder" window.
- Click the Start and Stop icons to start, resume or stop the debugging process.
- To stop the process at a certain point, insert a breakpoint in the test case. Right mouse click on a stencil and select **Toggle Breakpoint**.

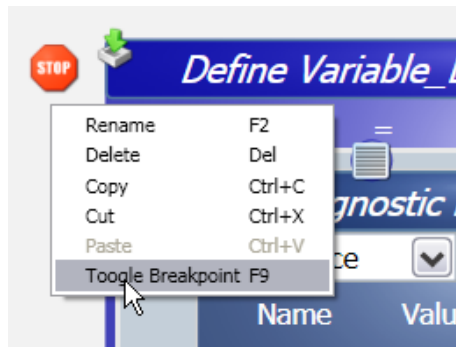


Figure 19: Toggle Breakpoint

To display the help texts

- Open the "Information Panel" window.



Figure 20: "Information Panel"

- Select the "Help" tab.
- Click in the list of categories on the stencil for which you need information. The left-hand side of the Information Panel shows the stencil, the right-hand side shows the associated information.

To display the messages

- Open the "Information Panel".
- Select the "Messages" tab.
- Double-click on the message for which you need information.
The stencil on which the message is based is highlighted within the sequence.
Note: This does not work for all messages, but only for those to which a stencil is related

To filter the messages

You can filter the message log according to the following types of messages:

- ALL
- INFO
- WARN
- ERROR

To do so, proceed as follows:

- Right mouse click anywhere in the "Messages" tab.
- In the context menu, select the **Filter** menu item and then the message type according to which you want the messages to be filtered.

Note: Each time the message log belonging to the sequence in the active tab will be displayed.

To copy a message

- Right mouse click anywhere in the "Messages" tab.
- Select **Copy**.
The information contained in the message (time stamp, type of message, etc.) will be stored in the clipboard.
You can now add the information using any text editor.

To reset the message log

- Right mouse click anywhere in the "Messages" tab.
- Select **Clear Log**.
The entire message log is deleted.

To close a sequence

- To close a sequence, select **File → Close Sequence**.
Or
Click the close button at the sequence tab header

2.3 Debugging

Debugging provides the possibility to insert a stop, 'Breakpoint', to stop the execution at the marked position and to continue it as soon as the play button or the buttons for stepping through the sequence are pressed. New sequences have to be saved before being able to debug the sequence.

Note: When the execution stops at a breakpoint, it stops before the execution of the marked stencil.

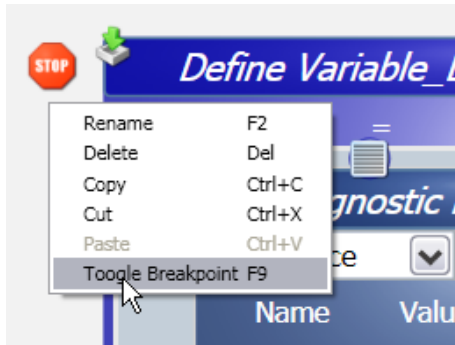


Figure 21: Toggle Breakpoint

A separate “Debug Messages” tab displays generated messages. When stepping through the sequence it shows the executed steps and the read values. The steps and values are not shown when running through the sequence without stopping. When you click on one of the steps, the focus of the sequence window is navigated to the related stencil.

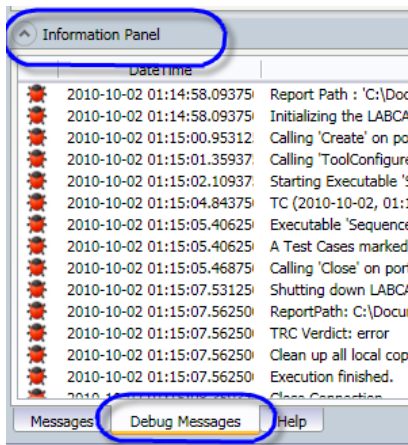


Figure 22: “Debug Messages” tab

You have buttons available at the top right position of the Automation Sequence Builder Window to start (▶) and stop (■) the debugger and it is possible to step into (▶■) the next stencil or after the current stencil (■▶) as well.

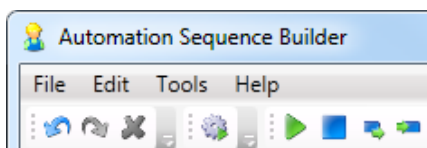


Figure 23: Toolbar

For debugging a test bench has to be used. The respective test bench configuration can be selected at the top of the Automation Sequence Builder Window.

To use this feature offline as well, you can select a real test bench at the top of the Automation Sequence Builder window and an offline test bench configuration will be generated automatically when starting the debugger. New options allow you to define a global test bench (frequently used, predefined bench configuration).

The debugging feature creates its own folders during execution, such as for the LABCAR-AUTOMATION project and for the created test bench.

Note: You need administrator rights to use this feature!

2.4 Hints for working with the ASB

2.4.1 Drag & Drop (Move vs. Copy)

The most convenient way to assemble and change Test sequences in ASB is by using Drag & Drop. Stencils are picked from one point and dropped to another.

When dragging stencils from the tool boy at the left or the favorites to the sequence or from the sequence to the favorites the drop will be performed as a copy action. When Drag & Drop stencils inside a sequence this will per default be a move action. If the intention is to copy the stencil this can be done by pressing the Ctrl key. The Ctrl key must be pressed after dragging the original stencil and still being hold when dropping at the intended target location.

In all cases the mouse cursor indicates if a move or copy action will be executed.

2.4.2 Drag & Drop 'highlighting'

You can move stencils from templates into the sequence pane or inside the sequence pane by drag & drop. Drag & Drop have a "highlight" effect showing where the drop action will occur.

This highlighting is available for the sequence pane and for the 'Favorites' pane as well.

2.4.3 The help and the result list

For a better overview; the result list, debug messages and the help pane are available as a tab in the "Information Panel". 'Messages' display all kind of messages (not a test case result) while 'Debug Messages' contains all messages of the debug run.

For both message lists it is possible to filter for 'Errors', 'Information' and 'Warnings' only. You can clear the log completely or copy one or more messages.

A very helpful feature is the possibility to navigate from a message in the "Messages" tab directly to the related stencil. This allows you to find easy and reliably the causing stencil, just by clicking the related message. The related stencil gets highlighted and focused then.

Resizing of the panel is possible via dragging the separation line.

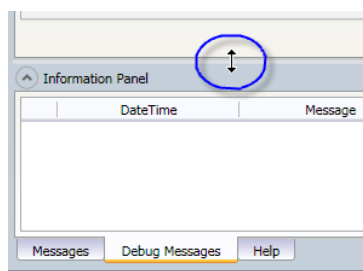


Figure 24: Separation line

Resizing of the single columns of the lists is possible as well by dragging the column separators.

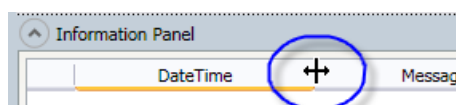


Figure 25: Resizing Columns

Note: When switching between different sequences the selected tabs of the message pane are kept in mind. Thus when returning to your previous sequence the same tab is shown which selected before for this sequence, independent which one was selected meanwhile for the others.

2.4.4 Error indications

When parameters in the stencils are missing, the stencil is not configured or defined correctly; the fields are highlighted by a red border to indicate the error in defining the parameters.

2.4.5 SuTMapping File

The path of the SuTMapping file is defined in the sequence's "Properties" dialog box. Select **File**→**Properties** to view the "Properties" dialog box.

You could also define the path of the SuTMapping file by selecting "Properties" from the context menu, accessible by right-clicking on the "Sequence" tab.

If no path is explicitly defined then the default file defined in the "Options" dialog box will be used. Select **Tools**→**Options** to view the "Options" dialog box.

There are two SuTMapping files; one for the model and the other for the ECU access ports.

2.4.6 Renaming Stencils

The names of the stencils can be modified to your convenience and to ensure the sequence is easily readable. To edit the name of the stencil, double-click the stencil and a "Rename" window is displayed. Include the name you want to provide to the stencil and click **Ok**.

2.4.7 Renaming variables

Beside stencils one might also want to rename variables (defined by variable stencils, like "Define Variable", "Define Curve Set" and "Define Diagnostic").

To do so two possibilities exist:

- By changing the variable name inside the VariableName combo box of a "Define ..." stencil, the name of only this variable stencil instance is changed. The only exception to this behavior is that also those "Use ..." stencils would be changed accordingly that reference to the same variable name, if no further "Define ..." stencils exist that also refer to that name.
- Within the context menu of "Define ..." stencils the entry "Rename All Variables" offers capability to change all "Define ..." and "Use ..." stencils referring to the variable name to change.

2.4.8 Generation of code out of the sequence

After creation of the test sequence there is the possibility to generate code out of it. There is only one generator that generates C# code. This is true for the "regular" test generator as well as for this library(DLL) generator. The code generator is part of the given template, thus, if another template is used, the different code generators may be present. With this version a C# code generator is provided initially.

2.4.9 Creation of different kinds of projects

The code which is created is based on different project compositions. The C# code itself is built into a Development project of Microsoft® Visual Studio® for further processing. The executable can be used for storage inside a global test release library only or for a complete LABCAR-AUTOMATION project, which can be created in this step as well.

2.4.10 Offline testing and Preparation of an offline test bench

Written sequences can be checked before release to a project or library. For this purpose an offline test possibility is provided by the Automation Sequence Builder.

The offline test bench which is used for offline testing can be saved and reapplied later on.

2.4.11 Stencil Compatibility

Icons are provided on the top of the stencils. These icons indicate the compatibility of the stencil. For example, Drag and drop a "Define Variable" stencil into the sequence editor. Notice the hand icon present on top of the "Value" field. This icon indicates that a stencil that contains a hand icon must be placed in the "Value" field. Hence for the explained example, you could drag and drop any of the stencils having a hand icon ("Constant", "Use Variable", etc). This concept hold good even for curves and diagnostics.



Figure 26: 'Hand' icon on top of the "Value" field

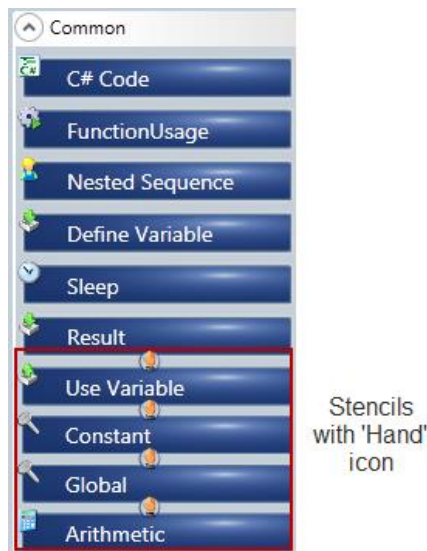


Figure 27: Stencils having 'Hand' icon

2.4.12 Hotkeys

The following table lists all hotkeys / shortcuts available in Automation Sequence Builder

Category	Name	ShortCut	Description	Menu
File Menu	New	Ctrl + N	To create a new sequence	Menubar
File Menu	Open	Ctrl + O	Opens a sequence saved previously (*.seq)	Menubar
File Menu	Save	Ctrl + S	Saves a sequence	Menubar
File Menu	Close Sequence	Ctrl + F4	Closes the active sequence	Menubar
File Menu	Exit	ALT + F4	Closes the Automation Sequence Builder	Menubar
Edit Menu	Undo	Ctrl + Z	Undoes the previous action	Menubar
Edit Menu	Redo	Ctrl + Y	Repeats an action undone	Menubar
Edit Menu	Cut	Ctrl + X	Cut the Stencil	Menubar / Context Menu

Edit Menu	Copy	Ctrl + C	Copy the Stencil	Menubar / Context Menu
Edit Menu	Paste	Ctrl + V	Paste the Stencil	Menubar / Context Menu
Edit Menu	Delete	Delete	Delete the Stencil	Menubar / Context Menu
Edit Menu	Rename	F2	Rename the Stencil	Menubar / Context Menu
Edit Menu	Find	Ctrl + F	Find Stencil	Menubar
Help	User´s Guide	F1	Opens the PDF file of the User´s Guide	Menubar
Debugging	Start	F5	Start debuggging	Toolbar
Debugging	Stop	Shift-F5	Stop Debugging	Toolbar
Debugging	Next	F10	Next Stencil	Toolbar
Debugging	Step Into	F11	Step into next Stencil	Toolbar
Debugging	Toogle Breakpoint	F9	Toogle Breakpoint	Toolbar / Context Menu
Navigation	Collapse/Expand All	Ctrl+Space	Collapse or Expand all Stencils in the Sequence or in a Group	Context Menu
Navigation	Collapse/Expand	Space	Collapse or Expand the selected Stencils	Context Menu
Navigation	Focus Sequence/Stencil	Up	Change the focus to the above stencil.	
Navigation	Focus Sequence/Stencil	Down	Change the focus to the below stencil.	
Navigation	Focus Sequence/Stencil	Pos1	The first stencil in the sequence get the focus.	
Navigation	Focus Sequence/Stencil	Ende	The last stencil in the sequence get teh focus.	
Navigation	Focus Sequence/Stencil	TAB	Step into next Stencil	
Stencils	<u>C</u> #_Code	Ctrl+Shift+E	Create Stencil	Context Menu
Stencils	<u>C</u> onstant	Ctrl+Shift+C	Create Stencil	Context Menu
Stencils	<u>G</u> lobal	Ctrl+Shift+B	Create Stencil	Context Menu
Stencils	<u>U</u> se Variable	Ctrl+Shift+U	Create Stencil	Context Menu
Stencils	<u>D</u> efine Variable	Ctrl+Shift+X	Create Stencil	Context Menu
Stencils	<u>R</u> esult	Ctrl+Shift+R	Create Stencil	Context Menu
Stencils	<u>A</u> rithmetic	Ctrl+Shift+A	Create Stencil	Context Menu
Stencils	<u>S</u> leep	Ctrl+Shift+S	Create Stencil	Context Menu
Stencils	<u>N</u> ested Sequence	Ctrl+Shift+Q	Create Stencil	Context Menu
Stencils	<u>C</u> omment	Ctrl+Shift+M	Create Stencil	Context Menu
Stencils	<u>L</u> ength	Ctrl+Shift+L	Create Stencil	Context Menu
Stencils	<u>W</u> hileDo	Ctrl+Shift+W	Create Stencil	Context Menu
Stencils	<u>L</u> oop	Ctrl+Shift+O	Create Stencil	Context Menu
Stencils	<u>G</u> roup	Ctrl+Shift+G	Create Stencil	Context Menu
Stencils	<u>D</u> oUntil	Ctrl+Shift+D	Create Stencil	Context Menu
Stencils	<u>I</u> f	Ctrl+Shift+I	Create Stencil	Context Menu
Stencils	<u>V</u> erify	Ctrl+Shift+V	Create Stencil	Context Menu

Stencils	Condition	Ctrl+Shift+N	Create Stencil	Context Menu
Stencils	Pass	Ctrl+Shift+P	Create Stencil	Context Menu
Stencils	Fail	Ctrl+Shift+F	Create Stencil	Context Menu
Stencils	LogicOperation	Ctrl+Shift+T	Create Stencil	Context Menu

2.4.13 Stencil Collapse/Expand All Hotkey (Ctrl + Space)

Collapse/Expand All option available in context menu of stencils collapses/expands all the stencils within the selected stencil (including selected stencil). Hotkey provided for this menu option (Ctrl + Space) is sometimes as well the default hotkey to switch between system languages. If that is the case, the system itself catches the hotkey and no application gets notified any more that this hotkey was used.

To use this available hotkey user can change/remove system hotkey from, Control Panel -> Region and Languages -> Keyboards and Languages tab (Change Keyboards button) -> Text Services and Input Languages -> Advanced key settings tab.

It can happen that even removing after removing the hotkey from the language settings Ctrl + Space still switches back to "Simplified Chinese". This is a bug in windows, that the UI for this setting does not remove the hotkey. In this case it only helps to uninstall the "Simplified Chinese Language".

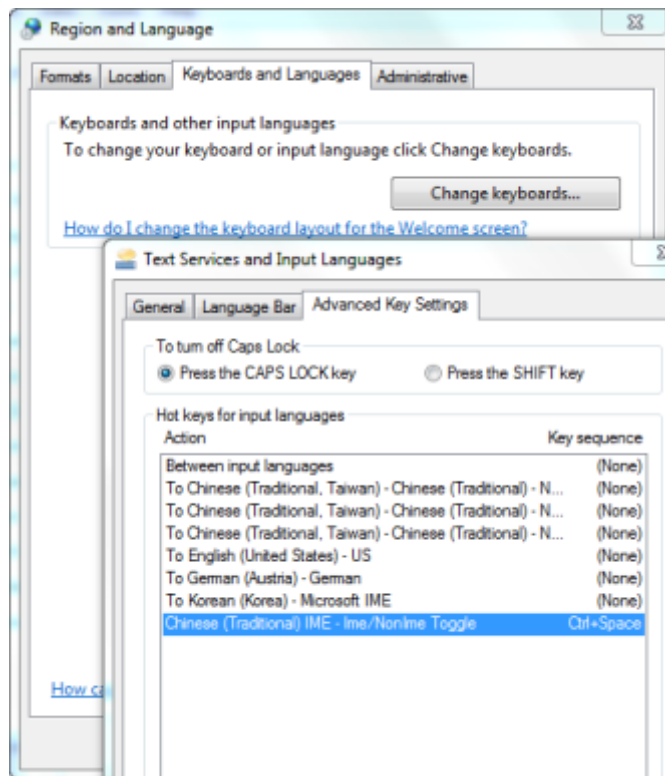


Figure 28: Change system hotkeys for changing the language.

3 Stencil Description

3.1 Common

Stencils that belong to the Common group provide a framework on which robust sequences are further built into.

3.1.1 Define Variable



Figure 29: "Define Variable" Stencil

The "Define Variable" stencil allows you to store values that can be reused later during test execution.

A new value assigned with one "Define Variable" stencil overwrites values assigned prior within the sequence.

Note: If the same name is used within multiple "Define Variable" stencils these stencils refer to the same variable.

Example

To work with a "Define Variable" stencil:

1. Drag and drop the "Define Variable" stencil in the sequence editor.
2. Type the variable name in the "VariableName" field.
 - Note:** Rename the stencil as described in section 2.4.6.
3. Drag and drop the relevant stencils such as "Use Variable", "Constant" that can be placed in the Define Variable stencil box. This defines the value to be stored.

3.1.2 Use Variable

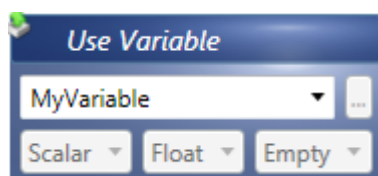


Figure 30: "Use Variable" Stencil

Values stored by variables can be retrieved by the "Use Variable" stencil.

The stencil's drop down list provides all variables defined within the sequence.

Note: An error is generated at runtime if you do not assign a value before using the variable

From the existing variables only a subset is provided depending on the value type (object type, data type, physical type) expected by the parent stencil of the "Use Variable".

For example; when a "Use Variable" stencil is placed as a child of the "Sleep" stencil, then all variables that are of physical type "Time" is displayed. In the case of the "Sleep" stencil, the corresponding physical type is 'Time'. However, the list not only provides those variables that exactly match the dimension that is expected by the parent stencil, but also those having a higher dimension such as arrays.

The "Sleep" stencil requires a scalar time value. However, while dropping a "Use Variable" stencil into it the dimensions that are offered are:

- a. scalar time variables
- b. arrays of time values

Note: The user must define the array values that must be used. Specify the array values and separate it with a ','.



Figure 31: "Use Variable" Stencil

Therefore the stencil provides an "Index" field, for example a "Constant" or "Use Variable" stencil can be dropped that defines the index. The "Index" field can be collapsed to reduce the stencil size as shown below.



Figure 32: "Use Variable" Stencil (Scalar)

Besides selecting a variable from the drop down list, this can also be done by using the "Variable Picker" that will appear when clicking the "..." button right next to the drop down list. The variable picker offers a filterable list of all variables that fit to the stencil requirement of the parent.

Example

To work with a "Use Variable":

1. Drag and drop the "Define Variable" stencil in the sequence editor. See Figure 31.
2. Type the variable name in the "VariableName" field. For example: TestVariable.
3. Drag and drop "UseVariable" stencil into relevant stencils that will accommodate it. For example: the "Sleep" or "Arithmetic" stencils. In this case, drag and drop the "Use Variable" stencil into the Sleep stencil.
4. Select the defined variable from the drop down list and define the index. For example: TestVariable. In this case, drag and drop the "Constant" stencil and define the value for the constant. The index values start from 0, 1, 2 etc. In this case, 0 maps to the array value 10, 1 maps to 12 and so on.
5. You can reduce the size of the stencil by collapsing the index access. See Figure 32.

3.1.3 C

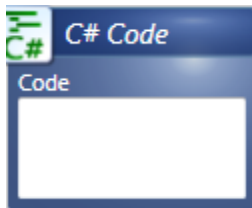


Figure 33: "C# Code" Stencil

The "C# Code" stencil includes the defined C# Code into the generated test case.

The given code is directly inserted into the generated test case code. You will need extensive knowledge of the generated code's structure to use this feature in a meaningful way and to not disrupt it.

Note: The "C# Code" stencil can be placed into each sequence field that is not indicated by a value icon.

3.1.4 Arithmetic



Figure 34: "Arithmetic" Stencil

The "Arithmetic" stencil provides capability to perform arithmetic operations (+, -, *, /). The operands have to be placed into the two value fields. Depending on the selected operation the operands must be of "compatible" physical as well as data types. For example; in case of

a '+' or '-' operation it would not make sense to define "1m + 1s". Hence for '+' and '-' the units must be of the same physical type such as "1m + 1m". In contrast "1m / 1s" would of course make sense.

ASB only knows those physical types and units that are defined in the PhysicalUnits.uct file present in the LCA's installation directory (a description on how these units have to be defined, refer to the comment file that is located in

```
C:\Program Files (x86)\ETAS\LABCAR-AUTOMATION
4.x\TestTools\Bin\PhysicalUnits.uct).
```

Note: If operations that are defined in ASB are not defined in that file this will be indicated by a red background of the two value fields and a corresponding tooltip.

Beside the operands and the operation to be applied to them also the desired target unit has to be selected from the lower drop down list of the stencil.

Note: Arithmetic stencil always returns a float value even if both the operands are integers. The only exception is when both operands are integer values and also of physical type "Empty".

Note: When adding operands to the stencil, the stencil displays only those operators that can be used with these operands for example: numbers can be +, -, /, *. Currently for all types other than scalar floats and integers no operators are implemented and therefore visible.

Note: The computation follows the [IEEE Standard for Floating-Point Arithmetic](#) (IEEE 754) which is used for the C# compiler. Division by zero gives an exact infinite result (\pm infinity).

Example

To perform Arithmetic operations:



Figure 35: "Arithmetic" Stencil Example

1. Drag and drop the Result stencil and then drop the Arithmetic stencil into it.
2. Drag and drop the Operand 1 stencil, select the arithmetic operations (+, -, *, /) and then the Operand 2 stencil into the requisite placeholders.

These operands can be "Constant", "Use Variable"- or again "Arithmetic" stencils. All stencils marked with a hand icon can be dropped into the respective placeholders of the operands.

Note: Only scalar values are supported.

3. Select the desired result's unit from the drop down list located at the left hand side of the "Arithmetic" stencil. For example; when dealing with length values (like 3km + 7m) it has to be defined if the result must be displayed as 'km' or 'm' (or mm, dm, ...).

3.1.5 Constant



Figure 36: "Constant" Stencil

The "Constant" stencil provides capability to define constant values of several types and units.

For example:

- dimension (Scalar, Array, Switch)
- data type (Float, String, Bool, ...)
- physical type (Length, Area, Volume, ...)
- unit (m, km, ms, s, ...)

Depending on the selection done within the stencil different control elements will appear that enables you to define values of the selected type.

For instance, if you select the dimension as "Switch", first the switch type must be specified, then the switch position must be selected from the provided options. The 'Switch' definition is read from the SwitchDefinitionTable specified in the **Tools** -> **Option** menu. The type of switch and the respective position has to be defined. For example, the switch "YesNoSwitch" with the possible values "Yes" and "No". Switches are used if there is only a defined set of valid values; for example, 'Low, Medium and High' <or> 'Pos1, Pos2, Pos3' <or> 'True and False'.

Whereas for numeric data types such as Integer, Float; a text box will appear to define the value. In contrast for a Boolean value a drop down box will appear. You can select appropriate values such as True and False.

Arrays have to be given with ';' as separator between the single values. For example '3.4; 4.7; 5' for a float array or 'abc; def; ghi' for a string array. Therefore the values must not include ';' and ' ' characters!

"Constant" stencils can be used within value fields of other stencils consuming the defined value (e.g. "Set Model Value" stencils to set a value in the model or "Sleep" stencil to pause the test execution for a duration defined by the constant).

3.1.6 Global

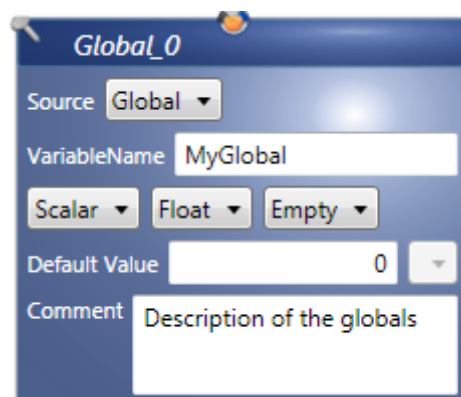


Figure 37: "Global" Stencil

The "Global" stencil provides access to parameters that are not defined in the current sequence itself but in a superior context. These can either be parameters defined with the

LCA project (in the Parameter Manager of the Test Manager) or within a parent sequence (that includes the current sequence with the Nested Sequence or FunctionUsage stencil).

Example

To add a global variable:

1. Drag and drop the "Global Variable" stencil into stencils defined by the hand icon.
2. Select Global or Sequence based on the Source option applicable for the program

Global: In case of a global parameter, it must be read from the parameter manager. The default value and the comment defined in the stencil while generating the test case, is written to the test case's .tpa file. With that the parameter is available within the Test Manager's Parameter Manager and the value can be changed accordingly.

Sequence: Parameters defined in a parent sequence can be retrieved in the same way as from the parameter manager. The only difference is on the definition side. The value has to be defined within the respective nested sequence call or function usage.

Moreover, the "child" sequence can also be executed solely (if generated with the "normal" test case generator). For more information see "Nested Sequence".

Note: If no value is defined outside (in the parent sequence resp. the Parameter Manager) the default value will be used.
3. Type the "VariableName" in the text field. To retrieve a parameter (from the project as well as from the parent sequence) its name has to be given in the stencil's text box. Further its dimension, physical type and unit has to be given like for the "Constant" stencil.
4. Set the relevant dimension, data type and physical type and unit using the drop down list
5. Assign a value in the "Default Value" field
6. You could include a comment as well, in the "Comment" field

3.1.7 Result



Figure 38: "Result" Stencil

The "Result" stencil is used to define the return value of a sequence. The stencil can be placed anywhere inside the sequence.

Note: The execution of the sequence is not aborted when a "Result" stencil is processed. The assigned value is passed to the parent sequence only after the sequence is processed completely. This also means that during the execution of a sequence several values can be assigned to the result variable. However, only the last assigned value is passed to the parent sequence.

For information on passing a result to the calling sequence (parent), refer to section 3.1.8, "Nested Sequence".

3.1.8 Nested Sequence

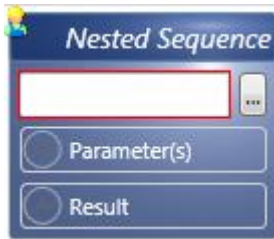


Figure 39: "Nested Sequence" Stencil

The "Nested Sequence" stencil is used to nest one or multiple sequences into another.

Example

Pre-requisite: To work with nested sequences, you must have an existing sequence created using the "Global" stencil. The parameters for the "Nested Sequence" are displayed only if the "Source" field of the "Global" stencils is set to "Sequence". For more information on passing parameters and the return value, see the relevant stencil description defined within "Global" and "Result".

See below:



Figure 40: Variables defined in 'getsum.asq' (Child sequence)

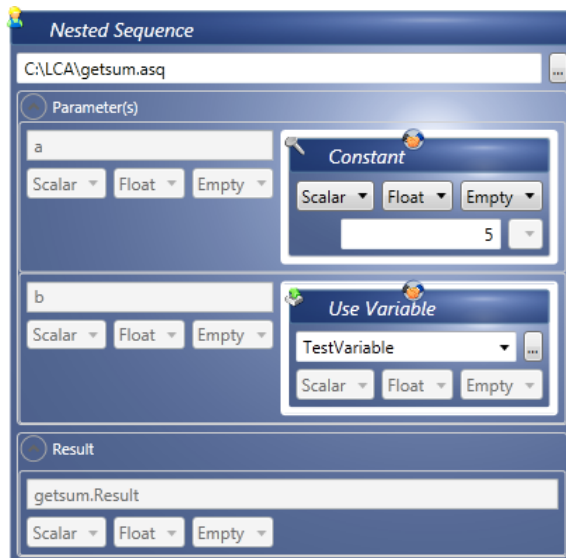


Figure 41: "Nested Sequence" Stencil Example (Parent sequence)

1. Drag and drop the "NestedSequence" stencil into the sequence editor
2. Click to choose the sequence file to be invoked. 'Sequence search dialog' window is displayed. You can browse through the directory tree (only those sequences within the current directory are displayed). The root path of the shown directory tree is determined by the path set in the tool options, see section 'Setting the Sequence Search Path' below.
The search path option is used to define a location where a sequence library is available.

Setting the Sequence Search Path:

- a. Click Tools -> Options. The Options window is displayed.
- b. In the 'Sequence Search Path (including Subfolders)' field, browse to the directory path. The path is used as a root path within the sequence search path dialog.

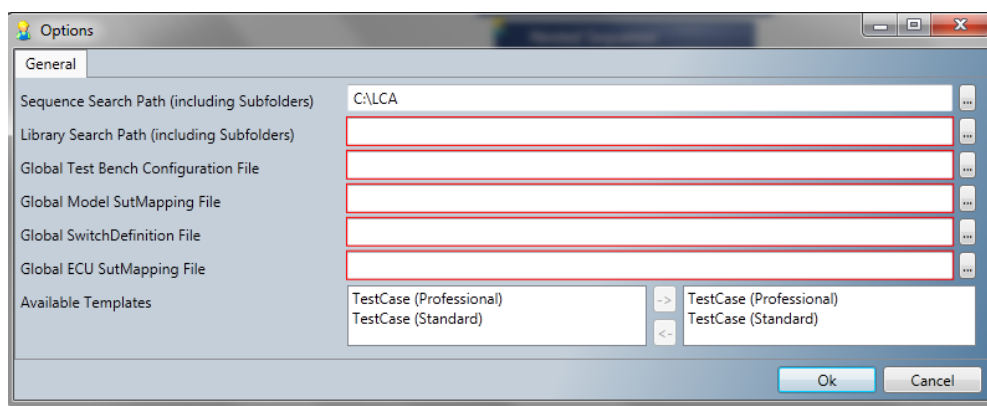


Figure 42: "Options" Dialog Box

This is the top most directory you will see when opening the search dialog button in the "NestedSequence" stencil. You can now navigate through that directory's sub-folders.

- c. You can select the respective filter, in this case, 'Filename' or 'Comment' or both in order to filter the search. For example: Type 'getsum' in the 'Filter' field and check both the options 'Filename' and 'Comment'. All files pertaining to getsum is displayed in the files section. To see the file details, you can select a respective file name. For example: getsum.asq

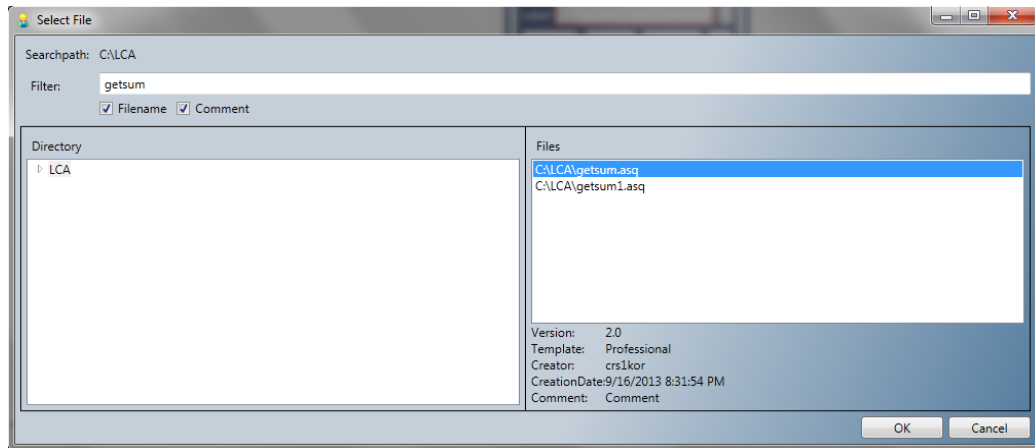


Figure 43: "Select File" Dialog Box

3. When selecting a sequence the stencil provides all the relevant parameters of the sequence as well as the return variable of the sequence. Parameters reflect the type information defined by the globals inside the parent sequence.

Note: Click 'Reload' if you have added a new parameter into the sub-sequence. In this case parameters 'a' and 'b' are automatically obtained from getsum.asq. If you add a new parameter 'c' in the getsum.asq sequence, then all you have to do is click 'Reload' in the parent sequence to include parameter 'c'.
4. Drag and Drop for example, a "Constant" stencil or a "Get Model Value Stencil" or a "Use Variable Stencil" to define the parameter values.
5. The return value will be available with the common "Use Variable" mechanism. In case of the above example, the "Use variable" stencil will provide a variable named "getsum.Result" providing the sub-sequence's return value.

The "Select File" dialog will not show the current sequence, even if it is contained in the search path. Adding the sequence itself as subsequence would lead to a recursion and the sequence could not be loaded again.

Note: It still is possible to create recursions with more than one sequence. E.g. "A uses B; B uses A"

3.1.9 FunctionUsage

The "FunctionUsage" stencil is similar to the "NestedSequence" stencil. Both are used to reuse functions defined in another sequence.

When a test executable is generated from the parent sequence, in background the functions of the sub-sequence are included into the parent sequence. The result is one executable that contains all the functions defined by the stencils out of the parent as well as of the sub-sequence.


Contrary to this, for the "FunctionUsage" at first a "Dynamic Linked Library (.dll)" is generated from the sub-sequence. This library is then linked to the parent sequence by help of the "FunctionUsage" stencil. When generating an executable from the parent sequence the functions out of the sub-sequence are not inserted into the executable directly. Instead only a link to the library is added. So during execution, the test executable will call the functions included in the .dll.

Example

Pre-requisite: To work with "FunctionUsage" sequences, you must have an existing sequence created using the Global stencil. The parameters for the "FunctionUsage" are displayed only if the "Source" field is set to "Sequence". For more information on passing

parameters and the return value, see the relevant stencil description defined within "Global" and "Result".

Generating the .dll file:

1. Create a sub-sequence. In this case, getsum.asq file as displayed in Figure 40 of Nested Sequence.
2. Click the 'Generate' icon. . The "Select Generators" window is displayed.
3. Select 'ATCLFunctionASBGenerator'.
4. Select the path where you want to save the .dll file and click 'Generate'. The '.dll' file is generated in the TRL folder of the sub-sequence. In this case, the sub-sequence is getsum.asq

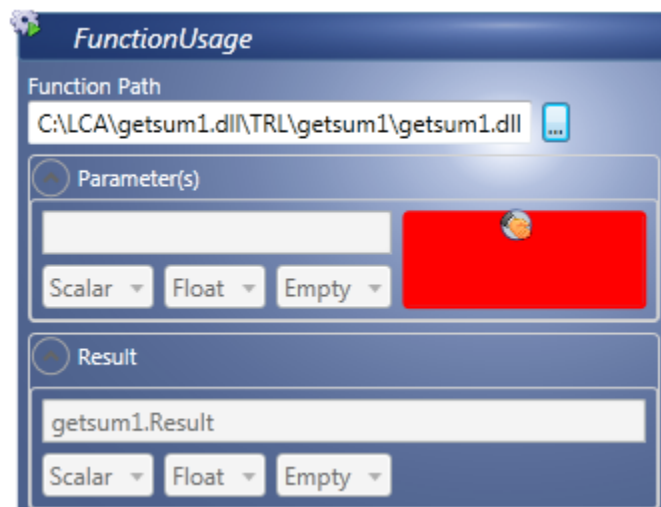


Figure 44: "FunctionUsage" Stencil

1. Drag and drop the "FunctionUsage" stencil into the sequence editor.
2. Browse for the sequence that you want to invoke. The '.dll' file is generated in the TRL folder of the sub-sequence. In this case, the sub-sequence is getsum.asq Result: Since you have defined the global stencil in getsum.asq, the parameters are automatically populated.
3. When selecting a sequence the stencil provides all the relevant parameters of the sequence as well as the return variable of the sequence. Parameters reflect the type information defined by the globals inside the parent sequence.
Note: Click 'Reload' if you have added a new parameter into the sub-sequence. In this case parameters 'a' and 'b' are automatically obtained from getsum.asq. If you add a new parameter 'c' in the getsum.asq sequence, then all you have to do is click 'Reload' in the parent sequence to include parameter 'c'.
4. Drag and Drop for example, a "Constant" stencil or a "Get Model Value Stencil" or a "Use Variable Stencil" into the respective value fields.
5. The return value will be available with the common "Use Variable" mechanism. In this case, the "Use variable" stencil will provide a variable named "getsum.Result" providing the sub-sequence's return value.

3.1.10 Sleep



Figure 45: "Sleep" Stencil

The "Sleep" stencil pauses the test execution for the duration defined by the child stencil. Add for example, a "Constant" or "Get Model Value" stencil as child to define the duration to be paused.

Note: The child stencil has to define a scalar float or integer value with physical type "Time". This scalar value can also be one out of an array, with the combination of an array and index.

3.2 Curve Evaluation

Curve Evaluation category of stencils are used to gather log values of signals which are used to create plots or graphical representation and to evaluate these curves.

3.2.1 Datalogger

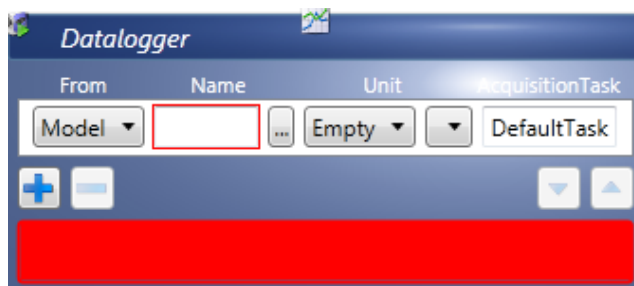


Figure 46: "Datalogger" Stencil

The "Datalogger" stencil logs values of ECU or Model for a set period. ECU datalogging logs the signals of EAM (ECU Access measurement port) and Model datalogging logs the signal of Model access port which is configured in the stencil's signal list.

For each signal; label, unit and the acquisition task has to be defined which is compared with the SuT mapping file provided with the test bench configuration. The label for the signal has to be of the defined physical type. If it is of a different unit of the same physical type a resp. conversion takes place for the logged values.

To define the log duration, either a sleep stencil is inserted in the "Datalogger" stencil list (to specify a predefined duration) or a loop with a suitable break condition. Either signal from Model or from ECU can be logged. In this case the Model resp. the EAM port is used. If signals of Model and ECU are defined within one datalogger stencil, they are logged by use of the SyncDL port (synchronous datalogger), which means the result will be a synchronous log of the two data sources. The time stamps are synchronized.

It is not possible to insert a "Datalogger" stencil into a "Datalogger" stencil, as only one can be executed at a point in time. If done, then it would lead to an error at runtime.

This should also be considered when adding a nested sequence or function call inside a

datalogger, as ASB cannot identify if a datalogger is used in that nested sequence or function call, this also leads to a runtime error.

Example

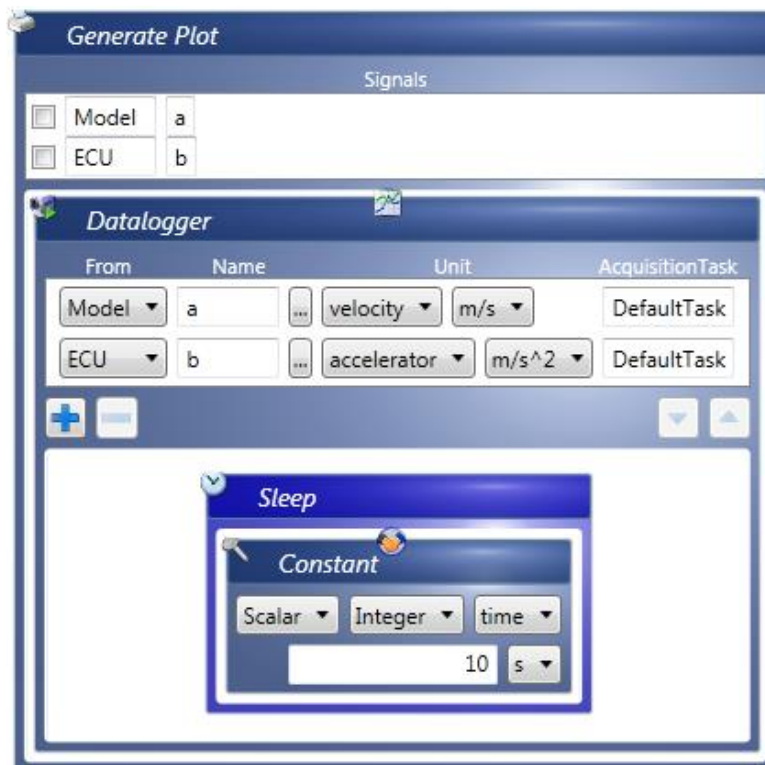


Figure 47: "Datalogger" Stencil Example


Pre-requisite: "Datalogger" stencil should be placed within a defined stencil

1. Drag and drop the "Generate Plot" stencil into the sequence editor
2. Drag and drop the "Datalogger" stencil into the "Generate Plot" stencil
3. Drag and drop the "Sleep" stencil into the "Datalogger" stencil
4. Drag and drop the "Constant" stencil into the "Sleep" stencil
5. Set time for 10 secs in the "Constant" stencil
6. In the "Datalogger" stencil set the "From" field to Model or ECU
7. Type Name, select the relevant Unit and the Acquisition Task
8. The signal values are logged for 10 secs

3.2.2 Generate Plot



Figure 48: "Generate Plot" Stencil

The "Generate Plot" stencil is used to plot graph for the selected curves onto the report. It is used with other stencils with the graph icon , for example: "Generate Plot" stencil is used in tandem with the "Datalogger" stencil.

Example

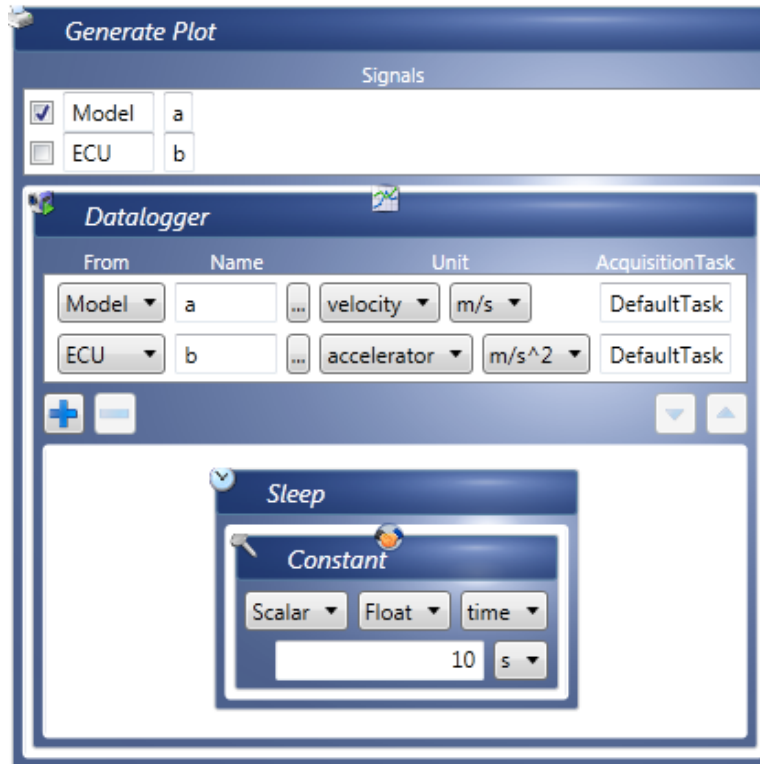


Figure 49: "Generate Plot" Stencil Example

Pre-requisite: "Generate Plot" stencil takes the curve values from the "Datalogger" stencil to plot graph.

1. Drag and drop the "Generate Plot" stencil into the sequence editor
2. Drag and drop the "Datalogger" stencil into the "Generate Plot" stencil
3. Drag and drop the "Sleep" stencil into the "Datalogger" stencil
4. Drag and drop the "Constant" stencil into the "Sleep" stencil
5. Set time for 10 secs in the "Constant" stencil
6. In the "Datalogger" stencil set the "From" field to Model or ECU
7. Type Name, select the relevant Unit and the Acquisition Task
8. The signal values are logged for 10 secs
9. As soon as the relevant values are selected in the "Datalogger" stencil, the entities are displayed in the "Generate Plot" stencil
10. Select the required entity or signal for plotting and run the sequence
11. The graph output is viewed in the report file

3.2.3 CurveValueComparision

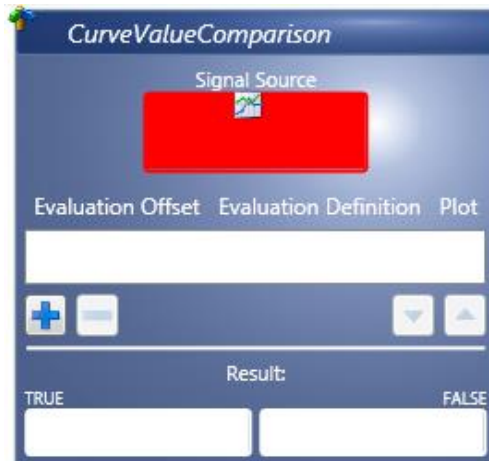


Figure 50: "CurveValueComparision" Stencil

The "CurveValueComparision" stencil is used to evaluate signals. "CurveValueComparision" or Curve sketching is a very powerful feature if an evaluation on curves has to be done. A study is done on the plotted curves by giving various values and finding out information on the behavior of the curves like when the curve signal reaches upward or positive and when it goes downward or negative or when it reaches threshold value inside a dedicated band.

The evaluation is based on the complete curve, starting from first measurement value, an offset or from the result time of the previous evaluation.

Each evaluation returns (internally) the time when the evaluation criterion is met. This time is used as start time for the next evaluation.

It is combined with additional offset and other attributes, depending on the evaluation formula selected.

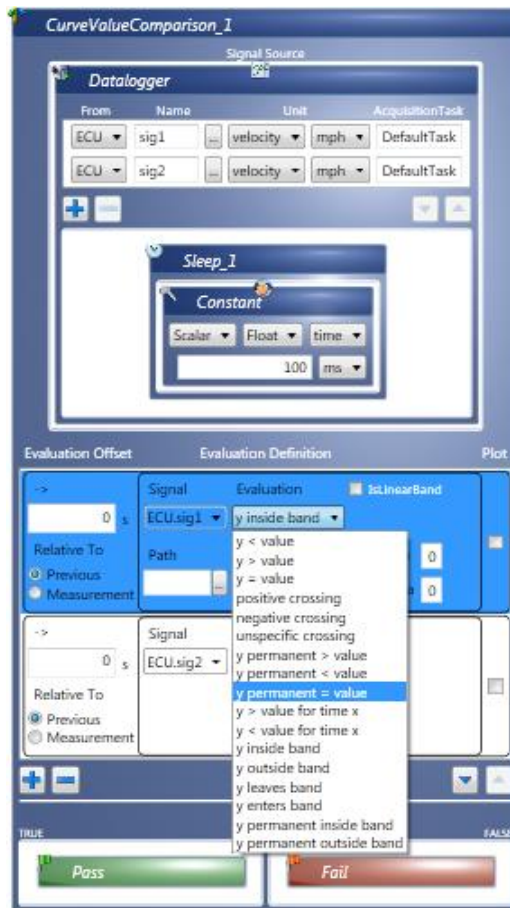
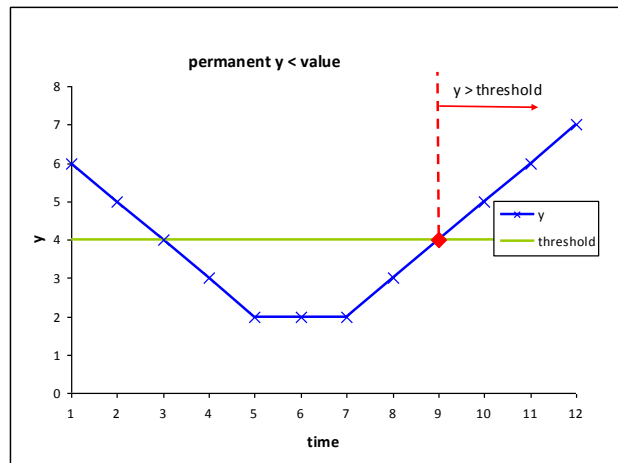


Figure 51: "CurveValueComparison" Stencil

Each evaluation can be added to the plot easily by just selecting a checkmark.

Evaluations using thresholds:

- Lower, higher or equal a dedicated value ($y < \text{value}$, $y > \text{value}$, $y = \text{value}$)
Returns the point in time (position at x-axis) when y is first time lower / higher or equal to the given value. For Example: if condition is $y < \text{value}$ and curve starts lower than the threshold value, the start point of evaluation is returned. If the curve starts higher than value, then the point in time is returned, when y is first time lower.
- Positive, negative, unspecific crossing
Returns the point in time (position at x-axis) when y crosses the given value upwards - from lower to higher range (positive crossing), downwards - from higher to lower range (negative crossing) or in either direction (unspecific crossing).
- Permanent lower, higher or equal dedicated value (permanent $y < \text{value}$, ...)
Returns the start (position at x-axis) when y is at start point lower than the threshold value and if it keeps lower threshold value for the complete time until logging end.
In contradiction to ' $y < \text{value}$ ' the time frame, when y was not yet lower the given value, is considered. If y starts lower than y (like condition checked) and becomes higher than the threshold till end of measurement, this check returns value and not a number.



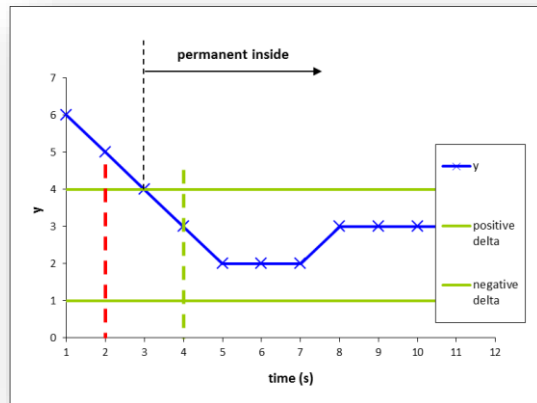
In the example, this function would (internally) return 'not a number', as it starts already higher than threshold value, while check 'y < value' would return x-position 3.

- Y lower or higher for time frame x
 - Acts like permanent lower/higher, but without starting absolutely from start point of measurement – it looks for the first point in time, when condition matched and check condition for the given time frame x. Returns the point in time when y is first time lower (or higher) then a given value. If the value does not hit this condition any time the value 'not a number' is returned.

Evaluations using bands:

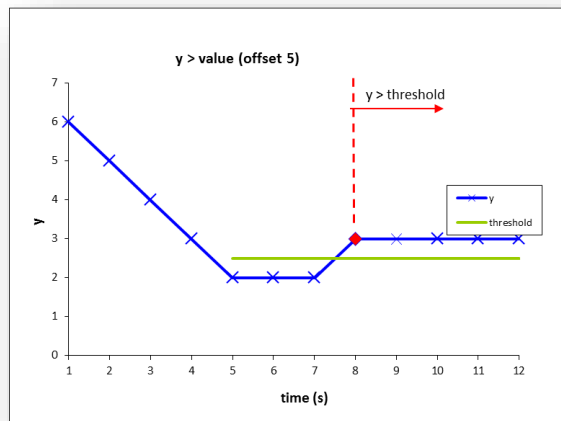
Bands represent a tolerance or a dedicated threshold value. They are defined as range of a positive and/or negative delta based on a given value. This value can be a fixed value or a curve.

- Inside, outside a band
 - Returns the point in time (position at x-axis) when y is inside the band first time (inside band) or outside first time (outside band).
- Leaves or enters a band
 - Returns the point in time (position at x-axis) when y leaves the tolerance band from inside or enters the band from outside first time.
- Permanent inside or outside a band
 - Returns the starting point of measurement in the x-axis when y is permanently inside (or outside) the given band or value 'not a number' if leaves (or enters) the band or starts already not inside (or not outside) the band. This function does consider the timeframe before the permanent-condition is met!



In the above example the function would (internally) return x-position 4, if measurement starts there, while it returns 'not a number' when measurement starts at x-position 2.

The threshold marker lines always start at the specified offset point only and not before. For Example: when defining a check that evaluates a signal exceeding a certain threshold at an offset of 3 against the measurement, the resp. evaluation line will begin at x-position 3.



In the example the evaluation starts at offset 5. Hence the first point of y that is greater than the threshold is at x-position 8.

Example

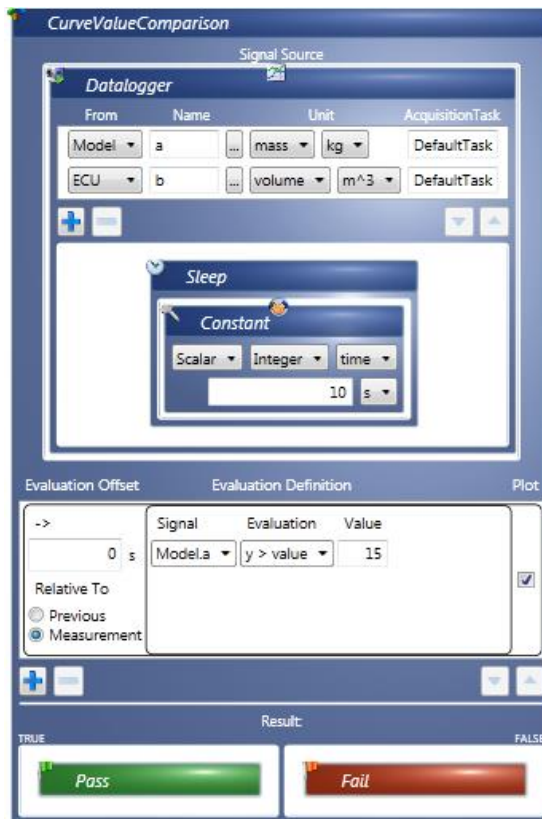


Figure 52: "CurveValueComparison" Stencil Example

1. Drag and drop the "CurveValueComparison" stencil into the sequence editor
2. Drag and drop the "Datalogger" stencil into the "CurveValueComparison" stencil
3. Drag and drop the "Sleep" stencil into the "Datalogger" stencil
4. Drag and drop the "Constant" stencil into the "Sleep" stencil
5. Set time for 10 secs in the "Constant" stencil
6. Once the "Datalogger" stencil values are set, the signal labels are available in "CurveValueComparison" stencil
7. Select "Signal" label from the drop down list
8. Select the relevant evaluation from the "Evaluation" drop down list like y>value, y<value, positive crossing, negative crossing and so on
9. Type the threshold value in the "Value" field
10. In the "Evaluation Offset" type offset time
11. Select "Previous" or "Measurement" radio button. If the "Previous" option is selected then the evaluation of the curve starts from the point in time when the last evaluation is returned. If the "Measurement" option is selected then the evaluation is carried out from the start
12. Depending on the result the sequence under "True" is executed else sequence under "False" is executed

3.2.4 Define Curve Set

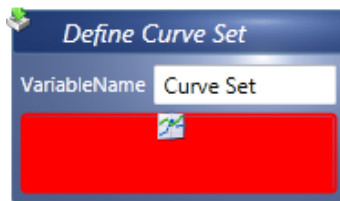


Figure 53: "Define Curve Set" Stencil

The "Define Curve Set" stencil is used to define a variable to store the curves or signals. The same signal log can be used at different points in the sequence by calling the variable.

Example

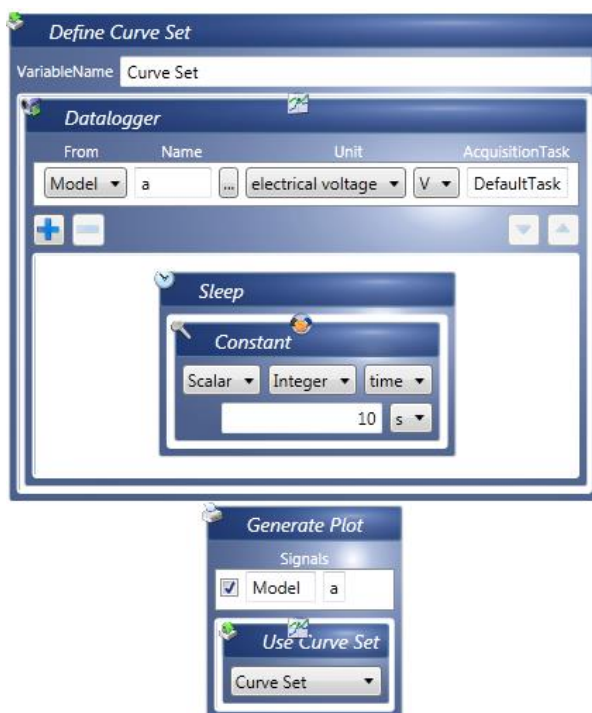


Figure 54: "Define Curve Set" Stencil Example

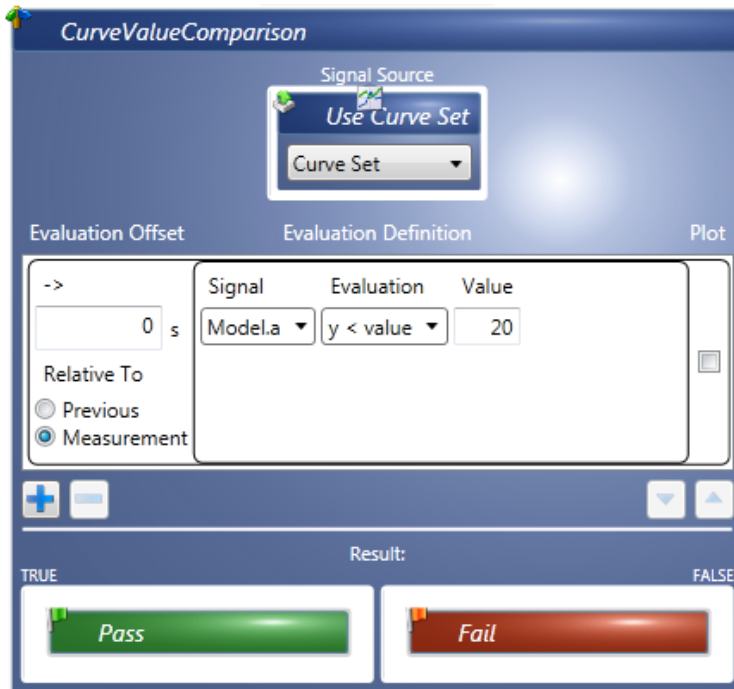



Figure 55: "CurveValueComparison" Stencil

1. Drag and drop the "Define Curve Set" stencil into the sequence editor
2. Type a variable name
3. Drag and drop stencil into the "Define Curve Set". The stencils having  icon can be used in the "Define Curve Set" stencil. In our example, the "Datalogger" stencil is used
4. The same defined curve set has been used throughout in the sequence and refers to the same signal or curve values

3.2.5 Use Curve Set



Figure 56: "Use Curve Set" Stencil

The "Use Curve Set" stencil is used in tandem with the "Define Curve Set" stencil. The defined curve set variable is accessed throughout the sequence using the "Use Curve Set" stencil.

Example

Please refer to the "Define Curve Set" stencil example.

3.3 Diagnostic

With the diagnosis stencils you are able use diagnosis services like reset the ECU (0x11) or test a service (0x25) and evaluate the results after it.

Note: The provided examples are dummies. The real services and its parameters have to be configured as described in the section 4.4.

3.3.1 Define Diagnostic

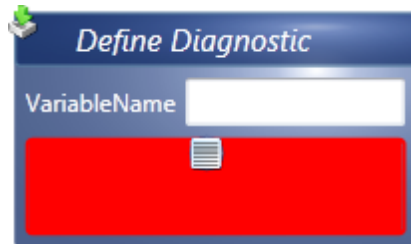


Figure 57: "Define Diagnostic" Stencil

"Define Diagnostic" stencil allows you to create a diagnostic variable that can be later referenced at multiple places.

The diagnostic variable is created by defining a variable name in the "VariableName" field and defining a value using a stencil with a diagnostic icon ("Use Diagnostic", "Diagnostic Diag").

The definition has to be placed before a later reference/usage in the sequence. When defining multiple variables with the same name the value is overwritten. Whenever the diagnostic variable is referenced, depending on its location, the nearest definition in front will be used.

Example

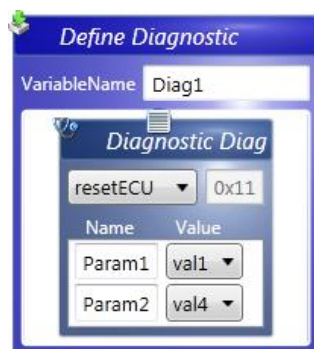


Figure 58: "Define Diagnostic" Stencil Example

The above example illustrates how you can define a diagnostic variable to reset an ECU.

1. Drag and drop a "Define Diagnostic" stencil into the sequence editor.
2. In the "VariableName" field, type 'Diag1'
3. Drag and drop a "Diagnostic Diag" stencil in the "Define Diagnostic" stencil
4. From the drop down box, select 'resetECU' and define the "Name" and "Value" of the parameters

3.3.2 Use Diagnostic



Figure 59: "Use Diagnostic" Stencil

"Use Diagnostic" stencil is used to choose the diagnostic variable to be referenced.

The drop-down list provides the list of diagnostic variables available. On selection of a diagnostic variable, the nearest definition of the variable, in front of this stencil will be used.

"Use Diagnostic" stencil is used with stencils that contain Diagnosis icon

Example



Figure 60: "Use Diagnostic" Stencil Example

The above example illustrates the use of "Use Diagnostic" stencil to reference a diagnostic variable for Diagnostic Evaluation.

Pre-requisite: A diagnostic variable must be defined using the "Define Diagnostic" stencil

1. Drag and drop a "DiagnosticEvaluation" stencil to the sequence editor.
2. Drag and drop the "Use Diagnostic" stencil into the "DiagnosticEvaluation" stencil
3. Use the drop down box to select a relevant diagnostic variable.
4. Based on the definition of the diagnostic variable, the "Check if:" field gets tabulated.

3.3.3 Diagnostic Diag

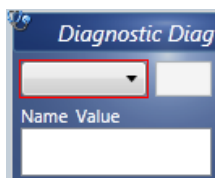


Figure 61: "Diagnostic Diag" Stencil

"Diagnostic Diag" stencil provides access to an ECU port for diagnosis. To configure the diagnostic services refer to section 4.4

Using the provided drop down box, the service to be queried can be defined. Depending on the service chosen, the service specific parameters with possible values are provided in the "Name" and "Value" lists. For each of the parameters, the desired value must be selected using the drop down box in the "Value" field.

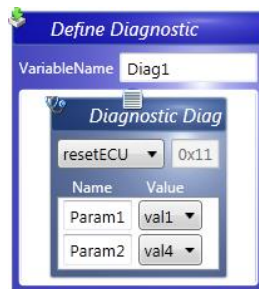
Example


Figure 62: "Diagnostic Diag" Stencil Example

The above example illustrates the usage of a "Diagnostic Diag" stencil to define a diagnostic variable to reset an ECU.

1. Drag and drop a "Define Diagnostic" stencil into the sequence editor.
2. In the "VariableName" field, type 'Diag1'
3. Drag and drop a "Diagnostic Diag" stencil in the "Define Diagnostic" stencil
4. From the drop down box, select 'resetECU'.
5. The list below "Name" and "Value" is updated.
6. Define the service parameters names in the "Name" field.
7. Select the required value, for each parameter, using the drop down box provided in the "Value" field.

3.3.4 DiagnosticEvaluation



Figure 63: "DiagnosticEvaluation" Stencil

"DiagnosticEvaluation" stencil allows you to evaluate the results of diagnosis services like reset the ECU (0x11) or test a service (0x25). These two services are only dummy examples available with the LCA/ASB installation. To have real ones available these have to be configured as defined in the section 4.4

When a diagnosis result is included in the top most control, a list box is generated in the "Check if:" field where a check on the result obtained can be defined. For each element of the diagnose result, you can define if the element should be considered for evaluation and (with the checkbox) and against which value it should be tested (with which operator).

If all selected evaluations are positive the sequence in the "TRUE" field will be executed, otherwise the "FALSE" field sequence will be executed. In most cases, setting of the verdict to Pass and Fail and is the only action, but the two fields can hold more actions as well or nothing if you like.

Example

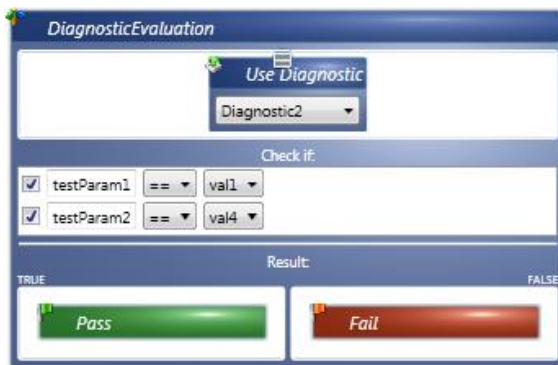


Figure 64: "DiagnosticEvaluation" Stencil Example

The above example illustrates the use of "DiagnosticEvaluation" stencil to evaluate the result of a diagnostic service.

Pre-requisite: A diagnostic variable must be defined using the "Define Diagnostic" stencil

1. Drag and drop a "DiagnosticEvaluation" stencil to the sequence editor.
2. Drag and drop the "Use Diagnostic" stencil into the "DiagnosticEvaluation" stencil
3. Use the drop down box to select a relevant diagnostic variable.
4. Based on the definition of the diagnostic variable, the "Check if:" field gets tabulated.
5. Select the checkbox to define which parameters must be considered for evaluation
6. Select the value against which the results have to be tested, using the drop down box
7. Drag and drop "Pass" stencil in the "TRUE" field and "Fail" stencil in the "FALSE" field.

The DiagnosticEvaluation stencil automatically writes the conditions to the report. This information is written as a table with the ReportLevel 50 so that it can be filtered in the reporting if desired. For the above example this could look like this:

Diagnostic Evaluation

Parameter	Value	Operator	Expected Value	Result
testParam1	Parameter 'testParam1' not found in the diagnostic result!	==	val1	false
testParam2	Parameter 'testParam2' not found in the diagnostic result!	==	val4	false

3.4 Documentation

"Documentation" is the category of stencils which are characterized by their ability to display text. This text is in the form of comments, warnings or error messages. These stencils are used in conjunction with other stencils or can be used separately. The output of report stencils are displayed in the report file whereas the "Comment" stencil adds comment to C# code generated from the test sequence.

3.4.1 Report Text



Figure 65: "Report Text" Stencil

“Report Text” stencil outputs the given text into the report file. The type of text can be “info”, “warning” or “error”.

Example

1. Drag and drop “Report Text” stencil into the sequence editor
2. Select the relevant option from the level drop down list
3. Result: The different results are displayed below:

Example 1:



1 Test Case : Sequence_2_1 (Sequence_2)

Test Result	
Test Result:	none
Start Time	9/17/2013 11:04:27 AM
Execution Time	00:00:00:483

.NET Runtime version 4.0.30319.18052

ATCL version 1.0.0.6

LABCAR-AUTOMATION version 4.1.0.40

Call Sequence f2a3efe2-43fc-4818-bfae-d6174e7a073f Location:Sequence_2

Call Stencil f309c0a1-2ebd-448a-bc39-56b2339cd8a1 Parameters Report: This is just an information.

This is just an information.

Exit Stencil f309c0a1-2ebd-448a-bc39-56b2339cd8a1 Parameters Report: This is just an information.

Exit Sequence f2a3efe2-43fc-4818-bfae-d6174e7a073f Verdict:none

Example 2:



1 Test Case : Sequence_2_1 (Sequence_2)

Test Result	
Test Result:	none
Start Time	9/17/2013 11:06:33 AM
Execution Time	00:00:00:474

.NET Runtime version 4.0.30319.18052

ATCL version 1.0.0.6

LABCAR-AUTOMATION version 4.1.0.40

Call Sequence f2a3efe2-43fc-4818-bfae-d6174e7a073f Location:Sequence_2

Call Stencil f309c0a1-2ebd-448a-bc39-56b2339cd8a1 Parameters Report: This is a warning.

WARNING: This is a warning.

Exit Stencil f309c0a1-2ebd-448a-bc39-56b2339cd8a1 Parameters Report: This is a warning.

Exit Sequence f2a3efe2-43fc-4818-bfae-d6174e7a073f Verdict:none

Example 3:



1 Test Case : Sequence_2_1 (Sequence_2)

Test Result	
Test Result:	none
Start Time	9/17/2013 11:10:32 AM
Execution Time	00:00:00:468

.NET Runtime version 4.0.30319.18052

ATCL version 1.0.0.6

LABCAR-AUTOMATION version 4.1.0.40

Call Sequence f2a3efe2-43fc-4818-bfae-d6174e7a073f Location:Sequence_2

Call Stencil f309c0a1-2ebd-448a-bc39-56b2339cd8a1 Parameters Report: This is an error message.

ERROR: This is an error message.

Exit Stencil f309c0a1-2ebd-448a-bc39-56b2339cd8a1 Parameters Report: This is an error message.

Exit Sequence f2a3efe2-43fc-4818-bfae-d6174e7a073f Verdict:none

3.4.2 Report Value



Figure 66: "Report Value" Stencil

"Report Value" stencil outputs the given text and value into the report file.

Example:

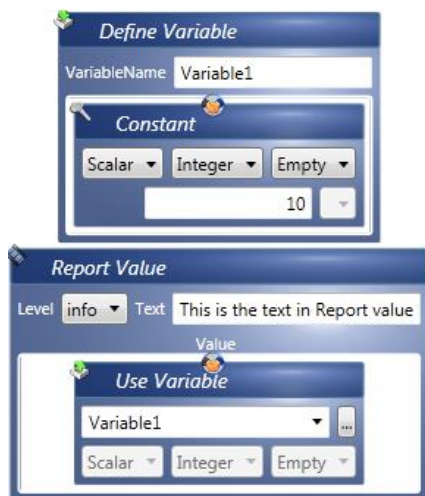


Figure 67: "Report Value" Example

1. Define a variable
2. Drag and drop the "Report Value" stencil into the sequence editor
3. Drag and drop relevant stencil into the "Report Value" stencil
4. When the Report value sequence is executed the output generated in the report is as shown in the image below

```

.NET Runtime version 4.0.30319.18052

ATCL version 1.0.0.6

LABCAR-AUTOMATION version 4.1.0.40

Call Sequence e2a3efe2-43fc-4818-bfae-d6174e7a073f Location:Sequence_2

Call Stencil 9d1f1fe1-0c6a-40d4-8b5a-a66267ab4e7f Parameters Label: Constant

Exit Stencil 9d1f1fe1-0c6a-40d4-8b5a-a66267ab4e7f Parameters Return Value : 10

Call Stencil 62407abf-2422-4c0f-99b9-ad2ba5840a19 Parameters Define Variable: Variable1

Exit Stencil 62407abf-2422-4c0f-99b9-ad2ba5840a19 Parameters

Call Stencil b7c4b620-e8c5-4779-9ce4-1a82b010938d Parameters Label: Variable1

Exit Stencil b7c4b620-e8c5-4779-9ce4-1a82b010938d Parameters Return Value : 10

Call Stencil 414c0dbe-eed3-4928-9d2f-da49ff9fc604 Parameters Report: This is th text in Report value Value: TypeSutInteger (Property:Value Value:10 Property:ValueMin Value:0 Property:ValueMax Value:0 Property:Unit Value:10 Property:Val Value:10 Property:Val_min Value:0 Property:Val_max Value:0 Property:Name Value:Variable1 Property:Lbl Value:Variable1 Property:Label Value:Variable1 Property:Comment Value: Property:IsLabelChangeable Value:FalseTestLabel changeable False )

Exit Stencil 414c0dbe-eed3-4928-9d2f-da49ff9fc604 Parameters Report: This is th text in Report value

Exit Sequence e2a3efe2-43fc-4818-bfae-d6174e7a073f Verdict:none
    
```

This is th text in Report value

Record:	comment	
	lbl	Variable1
	name	Variable1
	unit	
	val	10
	val_max	0
	val_min	0

Figure 68: "Report Value" Stencil Output in a Report

3.4.3 Comment



Figure 69: "Comment" Stencil

"Comment" stencil outputs the text as comment to the generated test case code.

3.5 Signal Generator:

The "Signal Generator" stencil provides a recorded signal sequence to a model input parameter to have realistic and reproducible conditions for tests. The signals must first be configured and then used.

3.5.1 Configure SG

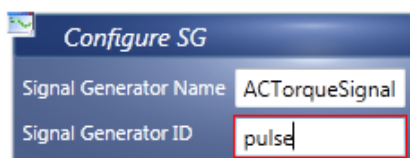


Figure 70: "Configure SG" Stencil

The signal generator is part of the model access port. Thus the configuration information is included in the model access Tool Configuration File. Within the Tool Configuration File (.tcf) for the model port, a StimuliInfo.xml file has to be given that specifies which signal generator out of the used model shall be used and which "data input" (the file containing the recorded data) shall be used.

For Signal generation you can have different set of input files. You can set the signal values at certain points of time in order to have reproducible test setups. This input file is a description of the signal. It can be created out of a recorded signal by LABCAR-OPERATOR (extension .sti) or edited by hand.

Note: The old available stimuli files with extension .lcs can be still used as well.

The input of this file will be assigned to a signal generator. The Signal Generator, running at the real time environment, builds up the signal and connects it to the signal input at the model. Thus the model uses this as signal feed instead of the output of a device or other model parts.

The data file can contain multiple sets of signal generators and data sets.

The set of information regarding signal to be used, acquisition task and file to be used are provided inside a stimuli set file additionally. This file will be automatically created by the configuration wizard at your test bench configuration folder at the subfolder \TBC and called per default StimuliSetInfo.xml.

A sample StimuliSetInfo.xml is provided below:

```
<?xml version="1.0" encoding="utf-8" ?>
<File type="Stimuli Set Info" extension="XML" version="1.0">
  <SG name="ACTorqueSignal">
    <SGModelName name="SignalGenerator_0" />
    <AcquisitionTask name="TaskDVEModel" />
    <SGSet name="pulse" file="TorqueOscillation.sti"/>
  </SG>
</File>
```

From the xml file, the information provided for SG name will be considered for "Signal Generator Name" and the information provided for SGSet name is considered for "Signal Generator ID" by the "Configure SG" stencil.

Note: The StimuliSetInfo.xml file created/provided by the Configuration Wizard is just an example. Therefore it will not contain "valid" information.

Therefore for a functional environment a valid file has to be created manually.

For more information on the test cases that are developed with respect to the Signal generator, see:

C:\Program Files (x86)\ETAS\LABCAR-AUTOMATION 4.x\Documentation\How To.pdf chapter 5.8 "Working with Signal Generator" and its sub-chapters.

Example:

To configure the signal:

1. Drag and drop the "Configure SG" stencil into the sequence editor
2. Type 'ACTorqueSignal' in the 'Signal generator Name' field. The Signal Generator Name is the same as the 'SG name' tag as specified in the StimuliSetInfo.xml file.
3. Type 'pulse' in the 'Signal Generator ID' field. The Signal Generator Id is the same as the 'SGSet' tag as specified in the StimuliSetInfo.xml file.

3.5.2 Action SG:



Figure 71: "Action SG" Stencil

The generation can be configured, started and stopped.

Example:

To work with the "Action SG" stencil:

1. Drag and drop the "Action SG" stencil into the sequence editor.
2. Select the relevant action from the drop down list 'Start', 'Stop' and 'Pause'.
 - 'Start' option starts playing the recorded values
 - 'Stop' option stops playing the recorded values
 - 'Pause' option pauses the playing of the recorded values
3. Select 'ACTorqueSignal' as the Signal Generator as it has been defined in the "Configure SG" stencil.

For more information on the test cases, see:

`\Users\Public\Documents\ETAS\LABCAR-AUTOMATION 4.x\Examples\Test Case Development\ASB_Sequences\TC_SignalGenerator_Evaluation.asq`

3.6 Fault Simulation

Fault Simulation FS category of stencils is used to define and activate electrical errors by use of fault simulation hardware.

3.6.1 Fault Simulation FS

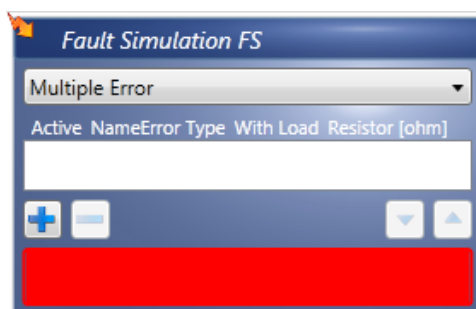


Figure 72: "Fault Simulation FS" Stencil

The "Fault Simulation" stencil is used to define and activate errors. The stencil provides a selection of different error categories, such as Multiple Errors, Fast Switch Errors, Fast Switch Resistor Errors and High Voltage Errors. The definition of the errors can be completed with a load flag (if applicable).



Mapping ASB error type to ES4440 error type

Category	ASB error type	ES4440 error type
Multiple Error	Open Load	
	Short Circuit To Potential	
	Short Circuit To UBatt	
	Short Circuit To GND	
Fast Switch Error	Open Load	
	Short Circuit To Potential	
	Short Circuit To UBatt	
	Short Circuit To GND	
Fast Switch Error Resistor Error	Pin To Pin Resistance	
	Leak Current To Potential	
	Leak Current To UBatt	
	Leak Current To GND	
	Inline Resistance	
High Voltage Error	Open Load	
	Short Circuit To Potential	
	Short Circuit	

Example



Figure 73: "Fault Simulation FS" Stencil Example

1. Drag and drop the "Fault Simulation FS" stencil into the sequence editor
2. From the drop down list choose the intended error category. Some of the error sets are "Fast Switch Resistor Error", "Multiple Error", "High Voltage Error" and so on
3. Click  to add new error types for a chosen error set
4. Based on the error category chosen the error types are displayed in the drop down list. Select the relevant error type
5. Type a name for the channel/pin for which the error has to be activated. This generally corresponds to the signal name of the PIN
6. Select the checkbox "With Load" if the error should be activated with connected load or uncheck if not.
7. Add the needed resistance value in ohms in the "Resistor" field which is needed to overcome the error
8. More than one error can be added by using 
9. To activate a particular error type select the "Active" checkbox
10. Any stencil like "Sleep", "Set Model value" can be added into the "Fault Simulation" stencil
11. In the example, "Sleep" stencil has been added with 10 secs as the value. The error is activated for 10 secs value of 2 is assigned to model parameter "test" and then exits

3.7 Model

"Model" category stencils are provided to allow the user to include model-related functions in the test sequence. By default, the .smf file mentioned in the 'Global Model SutMapping File' field that is given in the **Tools** -> **Options** menu is used when creating a sequence. This can be overridden for a specific sequence later. In order to specify the sequence specific file, refer to section 2.4.5

3.7.1 Set Model Value

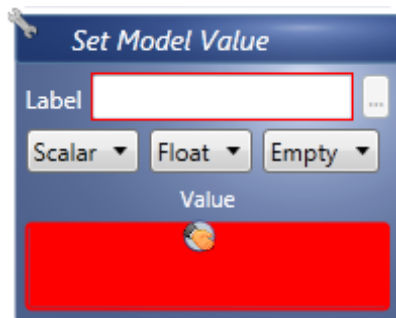


Figure 74: "Set Model Value" Stencil

The "Set Model Value" stencil allows you to set a value to a model parameter.

The model parameter label can be set either manually or with the help of the "Label Picker".

Manually you can set the parameter label by defining it using the stencil's control elements. The label name has to be specified in the "Label" field. The relevant dimension (Scalar, Array, Switch), data type (Float, Integer, String, Boolean) and physical type (Length, Area, Volume, ...) of the label have to be set using the provided drop down boxes.

You can also select the relevant parameter label with the help of the "Label Picker" that appears when clicking the "..."-button. The "Label Picker" will provide all labels present in the SuTMapping file given for the current sequence. On selecting the relevant label to use, all the other information is included automatically.

The value to be set has to be defined by placing an appropriate value stencil inside the "Set Model Value" stencil's "Value" field (e.g. "Constant" stencil, "Get Model Value" stencil).

In order to specify the SuTMapping file, refer to section 2.4.5.

Note: Model label can be defined manually or by using the label picker.

Example

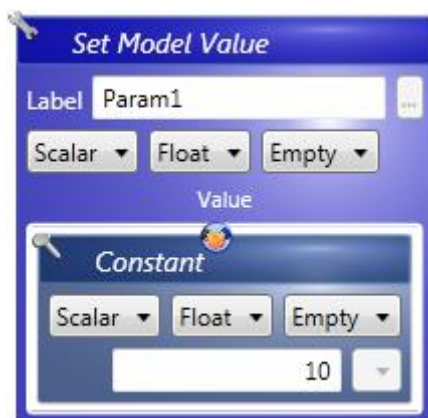


Figure 75: "Set Model Value" Stencil Example

The above example illustrates how you can manually set up a value for a model parameter.

1. Drag and drop the "Set Model Value" stencil in the sequence editor
2. Manually enter the parameter label in the "Label" field
3. Set the relevant dimension, data type and physical type using the drop down box
4. Drag and drop the "Constant" stencil in the "Value" field
5. Set the relevant dimension, data type and physical type for the constant
6. Type the value in the "Value" field

3.7.2 Get Model Value



Figure 76: "Get Model Value" Stencil

The "Get Model Value" stencil is used to retrieve the values of a model measurement or parameter.

The model's measurement or parameter, whose value must be retrieved, can be set either manually or with the help of the "Label Picker".

The measurement or parameter can be set by manually defining the right label using the stencil's control elements. Define the label's name in the "Label" field. Select the relevant dimension (Scalar, Array, Switch), data type (Float, Integer, String, Boolean), physical type (Length, Area, Volume, ...) and unit (ms, s, m, km, ...) using the drop down box. If the label is defined with different units when compared to the units defined in the SuTMapping file, but having the same physical type, the retrieved value will automatically be converted to the units defined in the stencil.

You can also select the relevant parameter label with the help of the "Label Picker" that appears when clicking the "..."-button. The "Label Picker" will provide all labels present in the SuTMapping file given for the current sequence. Also, only those labels that fit the requirements of the parent stencil are displayed. The labels provided exactly match the dimension or have higher dimension. On selecting the relevant label to use, all the other information is included automatically.

In order to specify the SuTMapping file, refer to section 2.4.5.

Note: Model label can be defined manually or by using the label picker.

Example

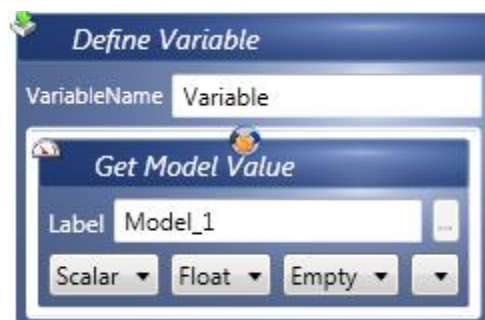


Figure 77: "Get Model Value" Stencil Example

The above example illustrates the usage of "Get Model Value" stencil to define a variable.

1. Drag and drop a "Define Variable" stencil into the sequence editor
2. Type the name of the variable as Variable in the "VariableName" field
3. Drag and drop the "Get Model value" stencil in the "Define Variable" stencil
4. Click the "..." button next to the "Label" field. Select the intended label from the list of labels displayed and model label along with its dimension, data type, physical type and unit is included in the stencil automatically

3.8 ECU

"ECU" category stencils are provided to allow the user to include ECU-related functions in the test sequence. Here the values are got from the ECU and not the Model. By default, the .smf file mentioned in the 'Global ECU SutMapping File' field that is given in the **Tools** -> **Options** menu is used when creating a sequence. This can be overridden for a specific sequence later. In order to specify the sequence specific file, refer to section 2.4.5

3.8.1 Set ECU Value

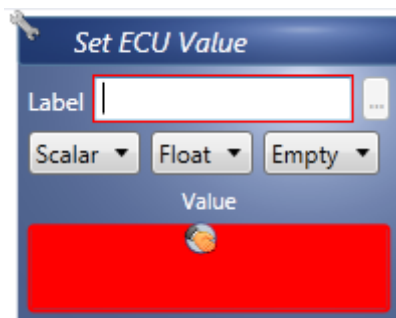


Figure 78: "Set ECU Value" Stencil

The "Set ECU Value" stencil allows you to set a value to an ECU parameter.

The ECU parameter label can be set either manually or with the help of the "Label Picker".

Manually you can set the parameter label by defining it using the stencil's control elements. The label name has to be specified in the "Label" field. The relevant dimension (Scalar, Array, Switch), data type (Float, Integer, String, Boolean) and physical type (Length, Area, Volume, ...) of the label have to be set using the provided drop down boxes.

You can also select the relevant parameter label with the help of the "Label Picker" that appears when clicking the "..."-button. The "Label Picker" will provide all labels present in the SuTMapping file given for the current sequence. On selecting the relevant label to use, all the other information is included automatically.

The value to be set has to be defined by placing an appropriate value stencil inside the "Set Model Value" stencil's "Value" field (e.g. "Constant" stencil, "Get Model Value" stencil).

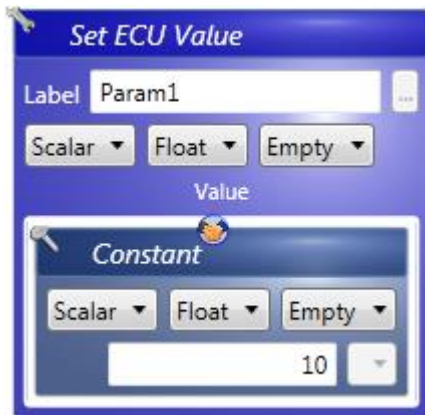
Example


Figure 79: "Set ECU Value" Stencil Example

The above example illustrates how you can manually set up a value for a ECU parameter.

1. Drag and drop the "Set ECU Value" stencil in the sequence editor
2. Manually enter the parameter label in the "Label" field. In this case, Param1.
3. Set the relevant dimension, data type and physical type using the drop down box
4. Drag and drop the "Constant" stencil in the "Value" field
5. Set the relevant dimension, data type and physical type for the constant
6. Type the value in the "Value" field

3.8.2 Get ECU Value

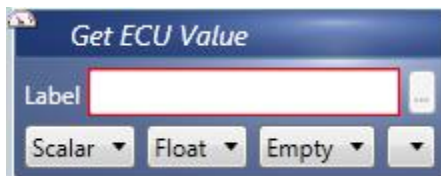


Figure 80: "Get ECU Value" Stencil

The "Get ECU Value" stencil returns a parameter value from the ECU.

The ECU's parameter, whose value must be retrieved, can be set either manually or with the help of the "Label Picker".

The parameter can be set by manually defining the right label using the stencil's control elements. Define the label's name in the "Label" field. Select the relevant dimension (Scalar, Array, Switch), data type (Float, Integer, String, Boolean), physical type (Length, Area, Volume, ...) and unit (ms, s, m, km, ...) using the drop down box.

You can also select the relevant parameter label with the help of the "Label Picker" that appears when clicking the "..." button. The "Label Picker" will provide all labels present in the SuTMapping file given for the current sequence. Also, only those labels that fit the requirements of the parent stencil are displayed. The labels provided exactly match the dimension or have higher dimension. On selecting the relevant label to use, all the other information is included automatically.

Example

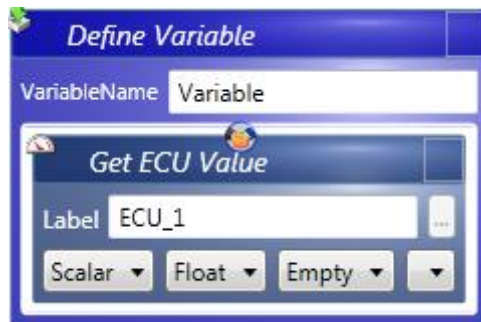


Figure 81: "Get ECU Value" Stencil Example

The above example illustrates the usage of "Get ECU Value" stencil to define a variable.

1. Drag and drop a "Define Variable" stencil into the sequence editor
2. Type the name of the variable as Variable in the "VariableName" field
3. Drag and drop the "Get ECU Value" stencil in the "Define Variable" stencil
4. Manually type the model label in the "Label" field and select the relevant dimension , data type, physical type and unit using the available drop down boxes

3.8.3 Get ECU Mea Value

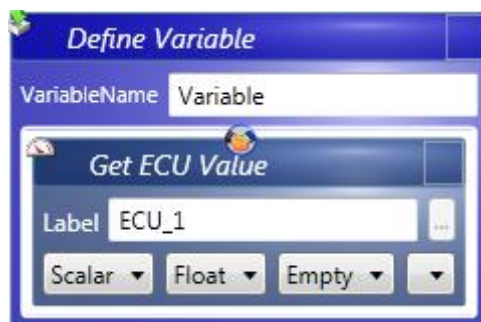


Figure 82: "Get ECU Mea Value" Stencil

The "Get ECU Mea Value" stencil is similar to the "Get ECU Value" stencil and returns a value from the ECU. However, the main difference between the stencil "Get ECU Value" and "Get ECU Mea Value" is that the stencil "Get ECU Mea Value" is used to get Measurement values (in contrast to parameter values) from the ECU.

3.9 Structures

"Structures" category consists of stencils that allow you to structure and condense large test case sequences. The stencils in this category allow you to make the sequence systematic and easily readable.

3.9.1 Group

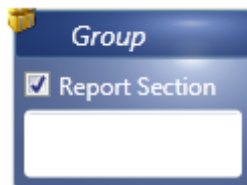


Figure 83: "Group" Stencil

"Group" stencil allows you to structure your test sequence to make it more readable and easier to maintain. You can define groups to clearly differentiate different portions of the test case. Groups can also be defined hierarchically.

Furthermore groups are easier to select, move, copy or designate as a favorite than individual stencils.

Select the "Report Section" checkbox to create a related section in the test case report.

Example

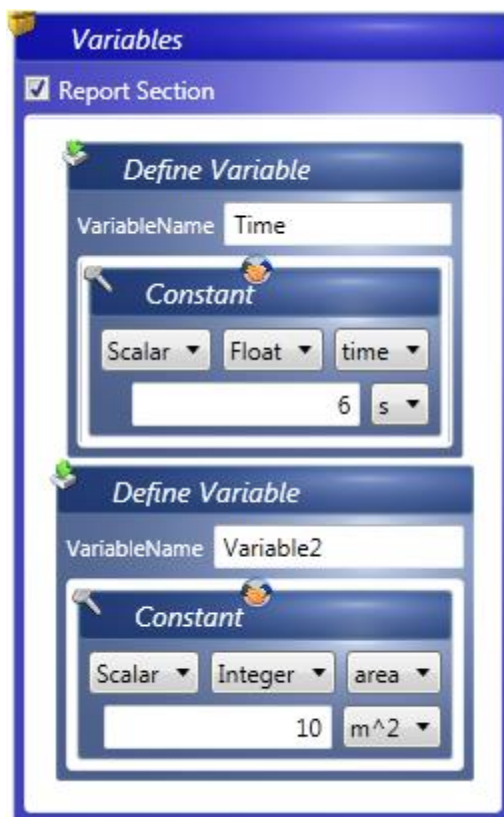


Figure 84: "Group" Stencil Example

The above example illustrates the use of a "Group" stencil to group all variables existing in the sequence.

1. Drag and drop a "Group" stencil into the sequence editor
2. Rename it to Variables by double clicking on the "Group" stencil
3. Drag and drop variables "Time" and "Variable2" defined using "Define Variable" stencil

3.9.2 Loop

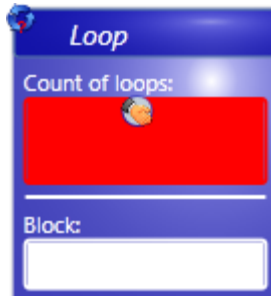


Figure 85: "Loop" Stencil

The "Loop" stencil allows you to define loops with a predefined number of iterations.

The number of iterations can be defined by any stencil that returns a scalar integer value with physical type set as "Empty" ("Constant", "Use Variable", "Arithmetic" etc)

The sequence to be looped is defined in the "Block" field.

Example

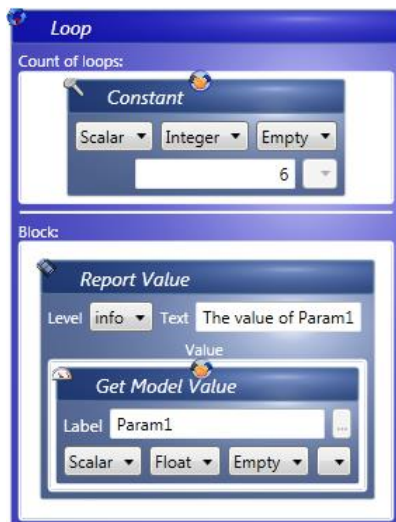


Figure 86: "Loop" Stencil Example

The above example illustrates the use of "Loop" stencil to report a model parameter value six times.

1. Drag and drop a "Loop" stencil into the sequence editor
2. Define the number of iterations to be carried out in "Count of loops:" field using a "Constant" stencil
3. Drag and drop a "Report Value" stencil in the "Block" field. Add the "Get Model Value" stencil in the stencil

3.9.3 WhileDo

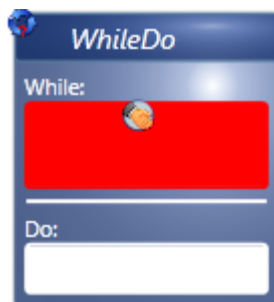


Figure 87: "WhileDo" Stencil

The "WhileDo" stencil allows you to define loops that are controlled by a condition. The condition can be defined by stencils that have a "Hand" symbol and return boolean values (boolean "Constant"s and "Variable"s, "Logic Operation"s, "Condition"s etc). The condition defined will be evaluated before each iteration. If the condition is not met, the following iteration will not take place. If the condition is not met for the first iteration, no iteration will be performed.

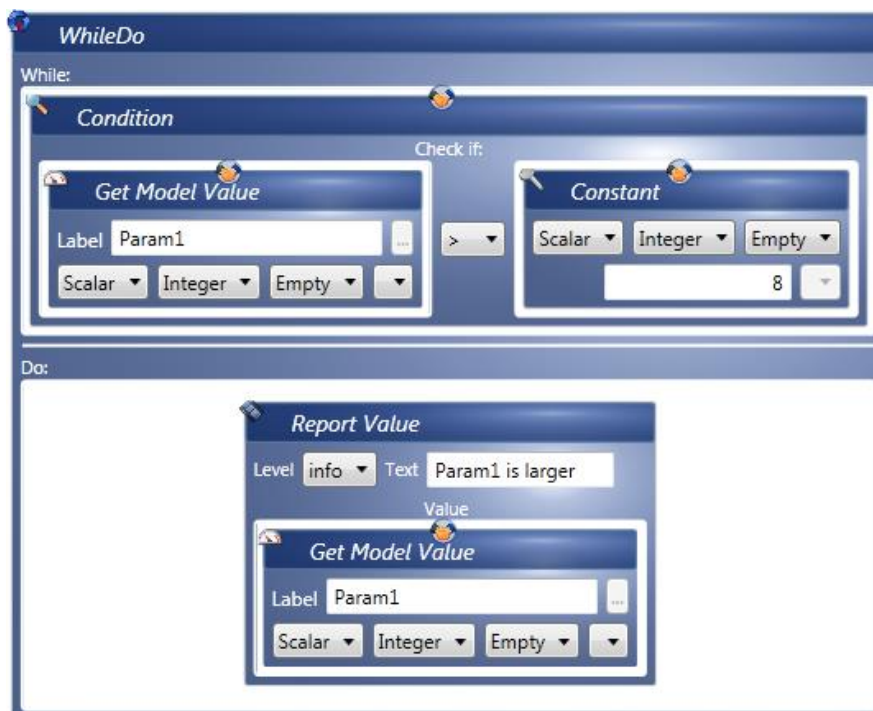
Example

Figure 88: "WhileDo" Stencil Example

The above example illustrates the use of a "WhileDo" stencil to report the value of a model parameter as long as the model value is greater than the defined constant.

1. Drag and drop the "WhileDo" stencil
2. Define the condition in the "While:" field using the "Get Model Value" and "Constant" stencil
3. Drag and drop the "Report Value" stencil to the "Do:" field

4. Drag and drop the "Get Model Value" stencil to the "Report Value" to define which value has to be recorded in the report. Also type the relevant info text in the "Text" field

3.9.4 DoUntil

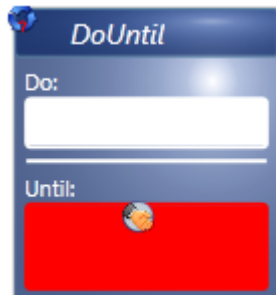


Figure 89: "DoUntil" Stencil

The "DoUntil" stencil allows you to define loops that are controlled by a condition. The condition can be defined by stencils that have a "Hand" symbol and return boolean values (boolean "Constant"s and "Variable"s, "Logic Operation"s, "Condition"s etc). The condition defined will be evaluated after each iteration. Compared to "WhileDo" stencil, at least a single iteration will be carried out.

Example

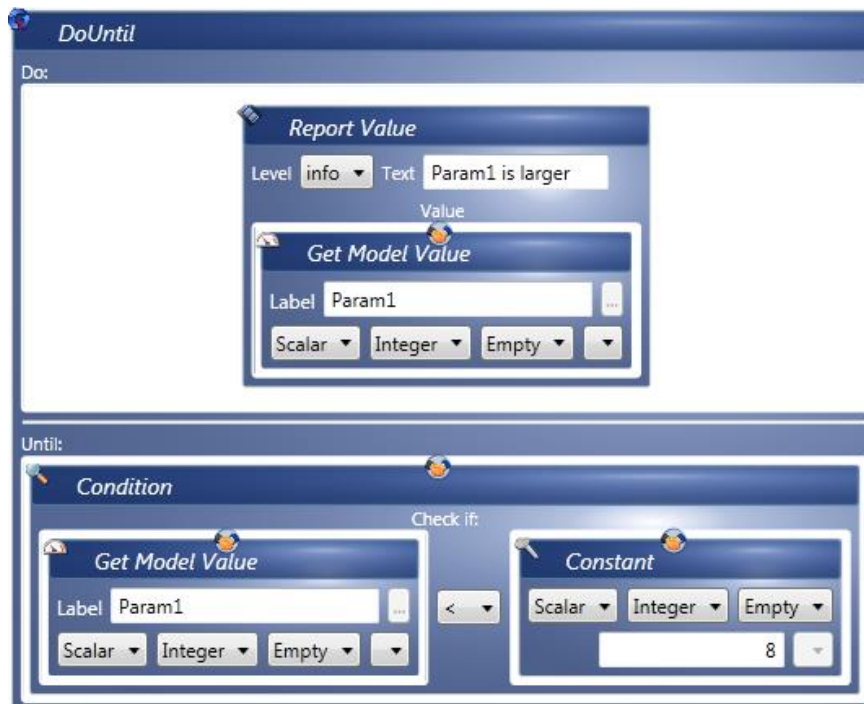


Figure 90: "DoUntil" Stencil Example

The above example illustrates the use of a "DoUntil" stencil to report the value of the model parameter, until the model value is less than the defined constant.

1. Drag and drop the "DoUntil" stencil
2. Drag and drop the "Report Value" stencil to the "Do:" field

3. Drag and drop the "Get Model Value" stencil to the "Report Value" to define which value has to be recorded in the report. Also type the relevant info text in the "Text" field
4. Define the condition in the "Until:" field using the "Get Model Value" and "Constant" stencil

3.9.5 Length

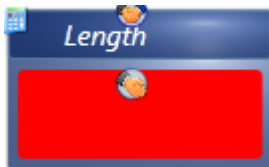


Figure 91: "Length" Stencil

The "Length" stencil allows you to determine the length of arrays and strings. This is helpful if you want to define a loop that iterates over all elements of an array or you want to use the string length to proceed further.

Example



Figure 92: "Length" Stencil Example

The above example shows how you can retrieve the array's length and afterwards construct a loop with that number of iterations.

1. Drag and drop a "Define Variable" stencil
2. Type the variable name as MyVariable in the "VariableName" field
3. Drag and drop the "Length" stencil into the "Define Variable" stencil
4. Drag and drop "Get Model Value" stencil into the "Length" stencil
5. Define the model label as MyArray and the data type as "Array"

Note: In the depicted example, the data type provided to the "Length" stencil has to be an array.

3.10 Verification

"Verification" category provides stencils which are used to check for conditions or verify conditions.

3.10.1 Condition



Figure 93: "Condition" Stencil

This "Condition" stencil is used for checking conditions. It cannot be used on its own and has to be used with another stencil. It is generally used with the "If" or "Verify" stencil. The operands have to be placed into the two value fields. Depending on the selected operation the operands must be of "compatible" types. For example: in case of a '+' or '-' operation it would not make sense to define "1m + 1s". Hence for '+' and '-' the units must be of the same physical type as well as data type such as "1m + 1m". In contrast "1m / 1s" would of course make sense.

The stencil will display operators in the operator list only those which can be used with the added operands and the operator list is left blank for the operands that are not implemented (like for arrays).

For example: If the operands use a variable which is of integer type then the operators displayed in the operator drop down list will be ">," "<," ">=," "<=," "==" and "! =". If the variable is of string type then the operator drop down list will display "==" and "! =". If no operands exist the drop down list is left empty.

The "Condition" stencil returns a scalar Boolean value.

Example:



Figure 94: "Condition" Stencil Example

Pre-requisite: "Condition" stencil should be placed within another stencil like the "If" or "Verify" stencil or any other stencil which expects a Boolean value.

1. Drag and drop the "Verify" stencil into the sequence editor
2. Drag and drop the "Condition" stencil into the Verify stencil field
3. Drag and drop relevant stencil into the operand fields. This can be "Constant", "Get Model Value" or the "Use Variable" stencils. This can be "Constant", "Get Model Value"

or the "Use Variable" stencils. In general, any stencil that returns a value (those with a hand icon on top) can be used in the value parameter. We can also have nested "Condition" stencils i.e., we can place "Condition" stencils in the value parameter. The stencils placed in the value parameters should be of the same data type, dimension and physical type

4. The value field is filled with the relevant stencil, if they are not compatible then the background remains in red color indicating that there is an error
5. The required operator is chosen from the comparison operator drop down list
6. **Result:** In the above example as the "Condition" stencil is used with the "Verify" stencil, if the condition is met the text in the Pass is displayed else the Fail text is displayed

3.10.2 Logic Operation

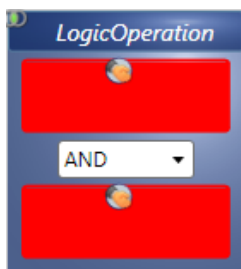


Figure 95: "Logic Operation" Stencil

This stencil is used to apply logical operation to two Boolean operands. This is not a standalone stencil i.e., it cannot be used by itself. It has to be used inside another stencil like the "If" or the "Verify" stencil.

Example:



Figure 96: "Logic Operation" Stencil Example

Pre-requisite: "Logic Operation" stencil should be placed within another stencil like the "If" or "Verify" stencil or any other stencil which expects a Boolean value.

1. Define two variables
2. Drag and drop the "If" stencil into the sequence editor
3. Drag and drop the "Logic Operation" stencil into the "If" stencil field
4. Drag and drop relevant stencil into the operand fields. The stencils in the operand field should return a Scalar Boolean value (e.g. "Constant", "Get Model Value", "Use Variable", "Condition" or another "Logic Operation" stencil)
5. Select the logical operators from the drop down list. The values being AND, OR and XOR.
6. **Result:** In the example, if the operation is successful "Pass" stencil is executed else the "Fail" stencil is executed

3.10.3 Pass



Figure 97: "Pass" Stencil

"Pass" stencil is used to specify the result of a successful test. (Refer example "Logic Operation stencil")

Note: If "Pass" or "Fail" stencils are incorporated in a group stencil, the "Fail" stencil takes precedence over the "Pass" stencil.

Example:



Figure 98: "Pass" Stencil Example

If a parent section has many child sections under it (as seen in the example above), even if one child section's output is a "Fail" then the parent section's output will also be a fail even if it has "Pass" from other child sections. If there is a "Fail" in the section the whole group will be affected and the whole section fails, as seen in the group section above with the "Pass" and "Fail" stencil, whereas the "Pass" stencil gets executed in the group section with only the "Pass" stencil present. Hence the "Fail" stencil has a higher precedence.

3.10.4 Fail



Figure 99: "Fail" Stencil

"Fail" stencil is used to specify the result of an unsuccessful test. In a section, the "Fail" stencil has a higher precedence over the "Pass" stencil. If a parent section has many child sections under it (refer "Pass" stencil example), even if one child section's output is a "Fail" then the parent section's output will also be a fail even if it has "Pass" from other child sections. (Refer example "Logic Operation stencil" and the "Pass" stencil).

3.10.5 If

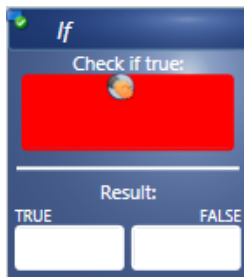


Figure 100: "If" Stencil

"If" stencil acts as a conditional statement in a test sequence. The condition is a Boolean expression. Depending on the result of the condition two different test sequences can be executed. If the result is "True" the test sequence under "True" is executed else the test sequence under "False" is executed.

Example:

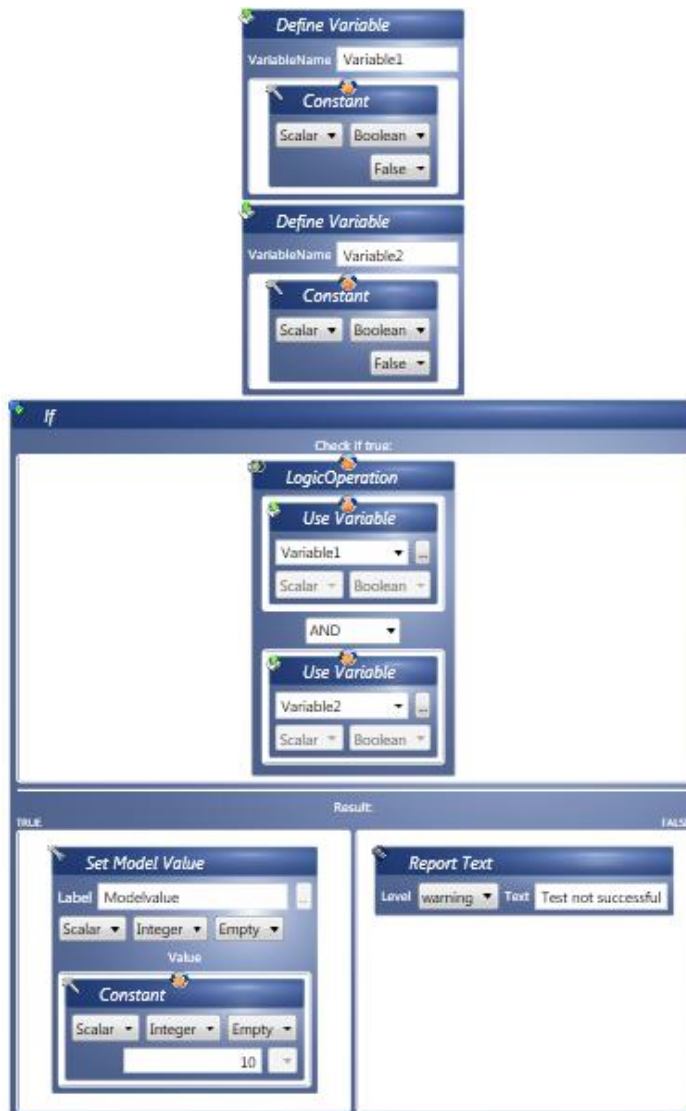


Figure 101: "If" Stencil Example

1. Define two variables
2. Drag and drop the "If" stencil into the sequence editor
3. Drag and drop relevant stencil into the "If" conditional field. The stencils which can be used are "UseVariable", "Condition" or "LogicOperation" stencil.

Note: When using the condition stencil based on the type of operands used the operators differ in the operator drop down list. For example: If the operands use a variable which is of integer type then the operators displayed in the operator drop down list will be ">," "<," ">=," "<=," "==" and "! =". If the variable is of string type then the operator drop down list will display "==" and "! =". If no operands exist the drop down list is left empty.

4. Complex sequences can be created by combining various logical expression in the "If" conditional field and combining the stencils together
5. The possible test sequence which has to be executed when the result of the "If" condition is True or False has to be incorporated in the True or False fields
6. Therein, all stencils can be used like in case of the "Group" stencil

- Result:** If the condition is met then the test sequence under True is executed else the test sequence under False is executed

3.10.6 Verify

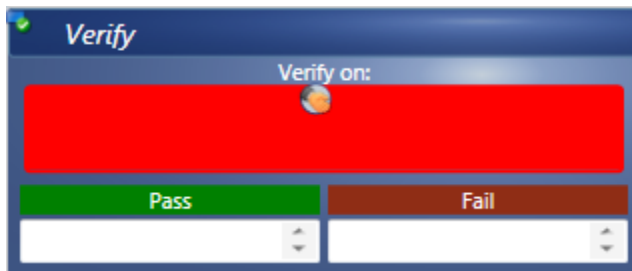


Figure 102: "Verify" Stencil

This stencil is used to verify values or expressions. It can be called as a convenient stencil, it is a combination of "If" stencil, "Report text" stencil and a verdict stencil. The example below shows how a "Verify" stencil can be used and also how an "If" stencil can be used to perform the same function a "Verify" stencil does.

Example:

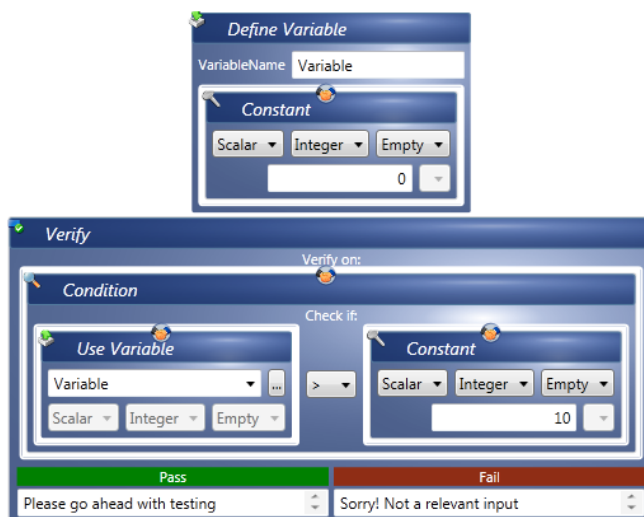


Figure 103: "Verify" Stencil Example

1. Define a variable
2. Drag and drop the "Verify" stencil into the sequence editor
3. Drag and drop the intended stencil into the "Verify" conditional field. This can be a "Use Variable" stencil, a "Condition" or a "Logic Operation"
4. Complex sequences can be created by combining these stencils together
5. Text is entered in the Pass or Fail parameter field which outputs the relevant result to report after the evaluation of the condition of the "Verify" stencil
6. The main difference between "If" and the "Verify" stencil is that in "If" stencil sequences are defined in the True or False field and one of them is executed but in Verify stencil the Pass or Fail field has text which is displayed in a report
7. **Result:** If the condition is met then the text in the Pass field is displayed else the text in the Fail field is displayed

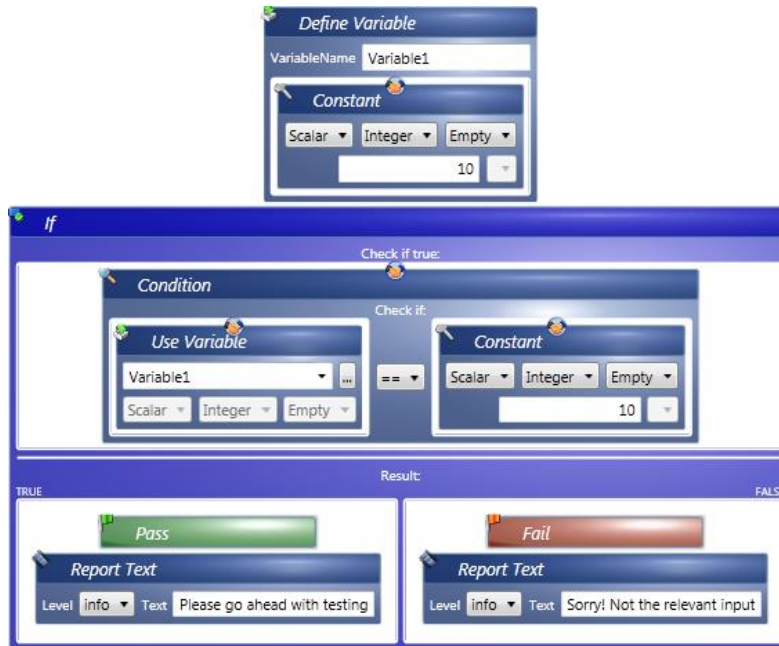


Figure 104: "Verify" Stencil usage depicted by "If" stencil Example

This example depicts how an "If" stencil can be used like a "Verify" stencil. The above example depicts the same flow and sequence like the "Verify" stencil example described earlier.

4 Use AUTOMATION SEQUENCE BUILDER with ODX-LINK 1.4

4.1 Prerequisites

To use the AUTOMATION SEQUENCE BUILDER with ODX-LINK 1.4 you need at least the LABCAR-AUTOMATION V4.1 Professional Package and the Tool-Adapter for diagnosis with ODX-LINK 1.4.

Note: ODX-LINK 1.4 runs with INCA 6.2.1.

Note: ODX-LINK 1.4.2 runs with INCA 6.2.1 and INCA 7.0

4.2 Test bench configuration

For your test bench configuration you have to add a model access "ModelAccess", an ECU access "ECUAccessMeasurement" and a diagnosis access "Diagnostics".

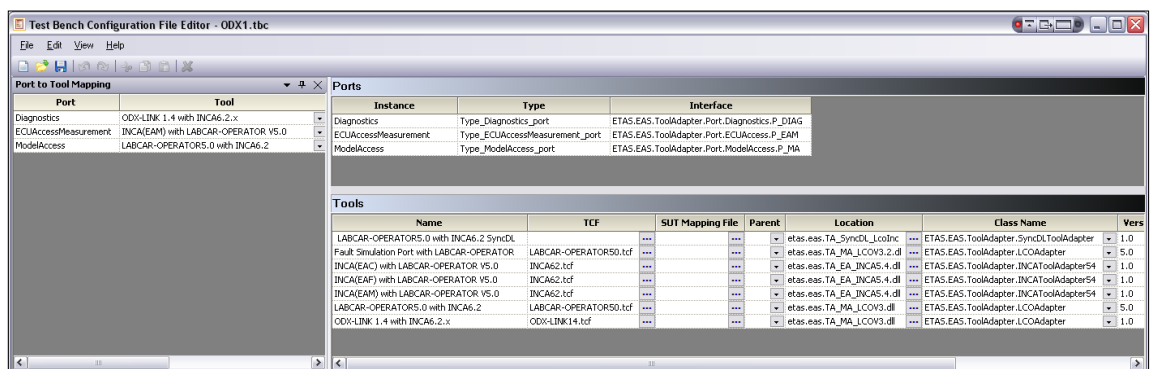


Figure 105: Test bench configuration for ODX-Link 1.4

4.3 Tool configuration

For use of the ODX-LINK 1.4 Tool-Adapter the tool must be configured.

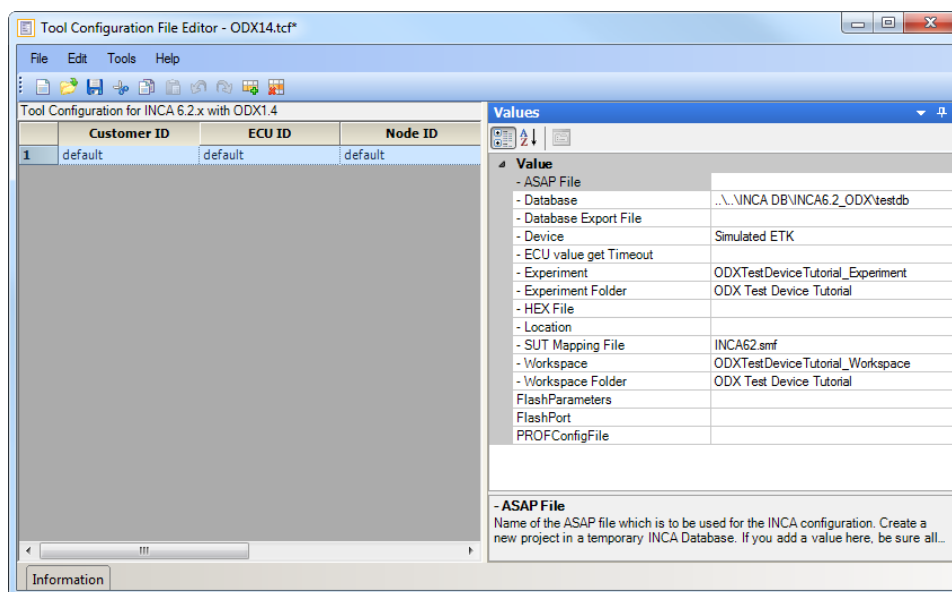


Figure 106: Tool configuration for ODX-LINK 1.4

The information you have to provide is the same which you use for your INCA Tool-Adapter. You have to refer to the INCA experiment that has included the ODX project you want to use.

Note: In conjunction to the Tool-Adapter for ODX-LINK 1.2 (in which the experiment is not allowed to have an ODX project included) the ODX project now has to be included in the experiment.

4.4 ODX services configuration

In case you are going to use the diagnosis port within an AUTOMATION SEQUENCE BUILDER (ASB) test case, you have to provide a Diagnostic Services configuration file.

In your LCA installation a small version is included for demonstration purpose. You can find it under:

```
C:\ProgramData\ETAS\LABCAR-AUTOMATION
4.x\conf\AutomationSequenceBuilder\DiagnosticServices.config
```

The configuration file is a XML file with a non-standard extension.

Structure of the XML file

The XML file starts with a standard XML header followed by list of objects. The first element in the objects container is a description.

```
<?xml version="1.0" encoding="utf-8"?>
<objects xmlns="http://www.springframework.net">
<description></description>
```

Next the objects for the diagnosis services follow.

For each service that shall be selectable in the diagnosis stencil in the AUTOMATION SEQUENCE BUILDER a separate object of type **DiagnosticServicesDataModel.DiagnosticService, DiagnosticServicesDataModel** has to be available.

```
+ <object id="resetECUService" type="DiagnosticServicesDataModel.DiagnosticService,
DiagnosticServicesDataModel">
+ <object id="testService" type="DiagnosticServicesDataModel.DiagnosticService,
DiagnosticServicesDataModel">
+ <object id="testService" type="DiagnosticServicesDataModel.DiagnosticService,
DiagnosticServicesDataModel">
```

The object itself needs the object **type** and a unique **id**.

```
<object id="resetECUService"
type="DiagnosticServicesDataModel.DiagnosticService,
DiagnosticServicesDataModel">
  <property name="Name" value="resetECU" />
  <property name="HexValue" value="0x11" />
  <property name="ParameterListExternal">
```

Each service object must have the properties *Name, HexValue, ParameterListExternal*. In addition if the service provides a *ReturnParameter* a *ReturnParameterList* property can be added.

```

<object id="testService"
  type="DiagnosticServicesDataModel.DiagnosticService,
  DiagnosticServicesDataModel">
  <property name="Name" value="testService" />
  <property name="HexValue" value="0x25" />
+ <property name="ParameterListExternal">
+ <property name="ReturnParameterList">
..</object>

```

While "HexValue" holds the hex-id of the service, "name" holds the name displayed in the dropdown list in the AUTOMATION SEQUENCE BUILDER.

The *ParameterListExternal* property holds a list with elements of type **DiagnosticServicesDataModel.ServiceParameter, DiagnosticServicesDataModel**

```

<property name="ParameterListExternal">
  <list element-
  type="DiagnosticServicesDataModel.ServiceParameter,
  DiagnosticServicesDataModel">
+ <object id="testParam1"
  type="DiagnosticServicesDataModel.ServiceParameter,
  DiagnosticServicesDataModel">
+ <object id="testParam2"
  type="DiagnosticServicesDataModel.ServiceParameter,
  DiagnosticServicesDataModel">
  </object>
  </list>
</property>

```

The elements of the list, each an object by itself, looks like

```

<object id="testParam1"
  type="DiagnosticServicesDataModel.ServiceParameter,
  DiagnosticServicesDataModel">
  <property name="Name" value="testParam1" />
+ <property name="ParameterValuesExternal">
</object>

```

The id of this service parameter *testParam1* has to be unique. The property name holds the value to be displayed in the dropdown in the ASB stencil.

The *ParameterValueExternal* property is a list holding elements of type **DiagnosticServicesDataModel.ServiceParameterValue, DiagnosticServicesDataModel**

```

<object id="testParam1"
  type="DiagnosticServicesDataModel.ServiceParameter,
  DiagnosticServicesDataModel">
  <property name="Name" value="testParam1" />
  <property name="ParameterValuesExternal">
    <list element-
  type="DiagnosticServicesDataModel.ServiceParameterValue,
  DiagnosticServicesDataModel">
</object>

```

While *ParameterListExternal* holds a list with the parameters provided by the service, *ParameterValueExternal* holds a list with the values allowed for each of the parameters.

```
<object id="testParam1"
type="DiagnosticServicesDataModel.ServiceParameter,
DiagnosticServicesDataModel">
    <property name="Name" value="testParam1" />
    <property name="ParameterValuesExternal">
        <list element-
type="DiagnosticServicesDataModel.ServiceParameterValue,
DiagnosticServicesDataModel">
            <object id="testParam1Value1"
type="DiagnosticServicesDataModel.ServiceParameterValue,
DiagnosticServicesDataModel">
                <property name="Name" value="val1" />
                <property name="HexValue" value="0x01" />
            </object>
        </list>
    </object>
</object>
```

Each object representing a *ServiceParameterValue* has an unique id, and a property providing a name and a value, that can be used by the ASB. The "HexValue" sent by the ECU is given in the next property. For each parameter value, a "Name" to be selectable in the dropdown menu and a "HexValue" to be sent is given.

The structure for the return parameters and values is similar.

It is a list holding objects representing *ServiceParameters*. Each object itself has a name and a list of parameter values as properties. The list of parameter values contains objects with a name to display in the dropdown of the diagnosis stencil and a "HexValue" to be received.

```
<property name="ReturnParameterList">
    <list element-
type="DiagnosticServicesDataModel.ServiceParameter,
DiagnosticServicesDataModel">
        <object id="ReturntestParam1"
type="DiagnosticServicesDataModel.ServiceParameter,
DiagnosticServicesDataModel">
            <property name="Name" value="testParam1" />
            <property name="ParameterValuesExternal">
                <list element-type="DiagnosticServicesDataModel.
ServiceParameterValue,
DiagnosticServicesDataModel">
                    <object id="ReturntestParam1Value1"
type="DiagnosticServicesDataModel.ServiceParameterValue,
DiagnosticServicesDataModel">
                        <property name="Name" value="val1" />
                        <property name="HexValue" value="0x01" />
                    </object>
                    + <object id="ReturntestParam1Value2"
type="DiagnosticServicesDataModel.ServiceParameterValue,
DiagnosticServicesDataModel">
                </list>
            </property>
```

```

</object>
+   <object id="ReturntestParam2"
      type="DiagnosticServicesDataModel.ServiceParameter,
      DiagnosticServicesDataModel">
</list>
</property>

```

4.5 Test case

In the example below (Figure 107) the service "readDTCByStatus" is used with the parameters "groupOfDTC" and "statusOfDTC". In the evaluation the return parameter "numberOfDTC" is chosen to be compared to an expected value of 2.

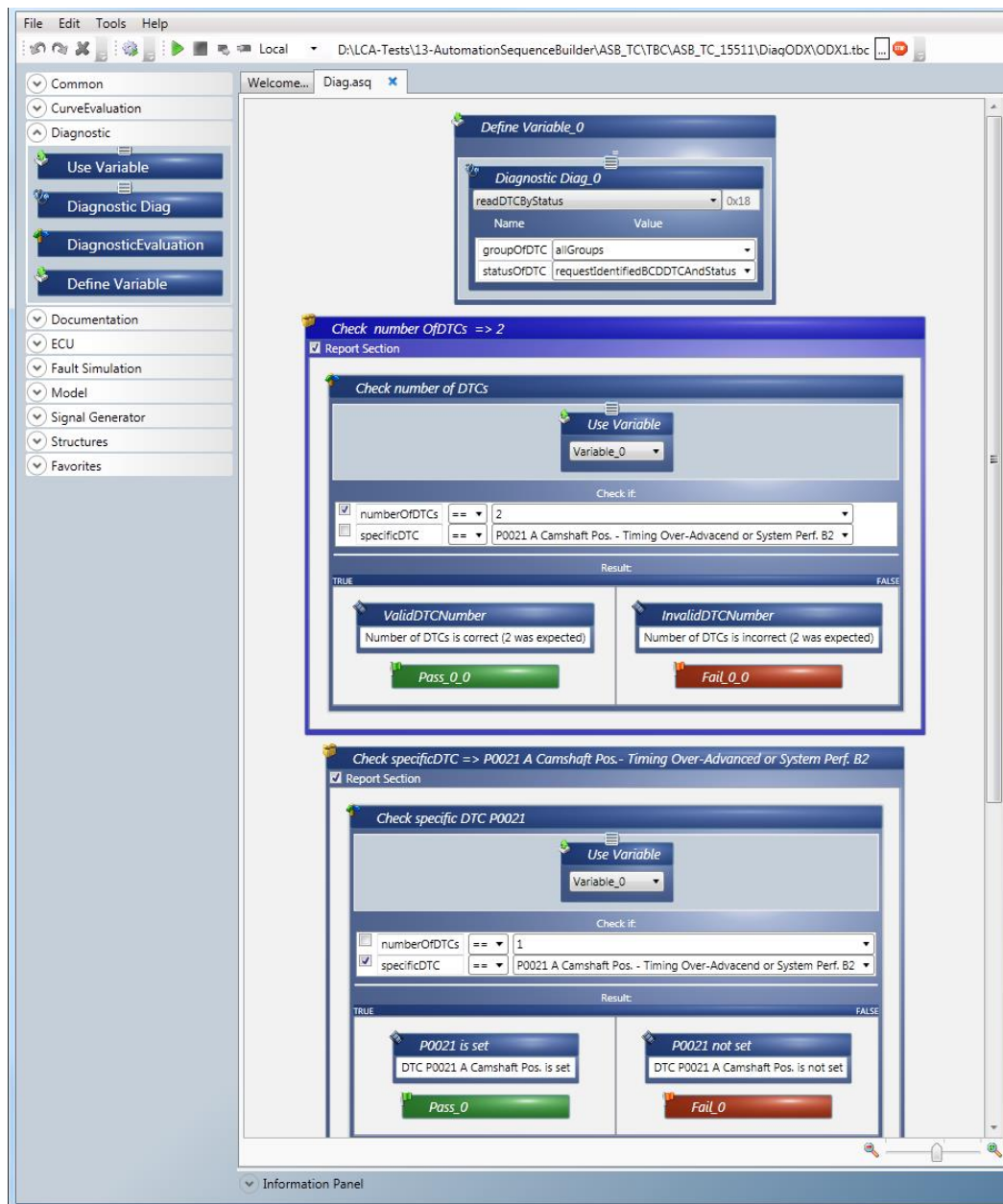


Figure 107: ASB test case using diagnosis stencils

Remember, the values selectable through the dropdown menu are given by the ODX services configuration (see section 4.4).

5 ETAS Contact Addresses

ETAS HQ

ETAS GmbH	Phone:	+49 711 3423-0
Borsigstraße 14	Fax:	+49 711 3423-2106
70469 Stuttgart	WWW:	www.etas.com
Germany		

ETAS Subsidiaries and Technical Support

For details of your local sales office as well as your local technical support team and product hotlines, take a look at the ETAS website:

ETAS subsidiaries	WWW:	www.etas.com/en/contact.php
ETAS technical support	WWW:	www.etas.com/en/hotlines.php

ETAS

■