DRIVING EMBEDDED EXCELLENCE

ETAS LABCAR-MODEL V4.3

User Guide

www.etas.com

Copyright

LABCAR-MODEL V4.3 - User Guide R01 EN - 12.2020

Contents

1 About this Document

1.1 Introduction

This document addresses qualified personnel working in the fields of automotive ECU development and testing. Specialized knowledge in the areas of embedded systems and simulation is required.

This document describes the handling and the licensing of the LABCAR-MODEL simulation models and contains the following sections:

- “LABCAR-MODEL Family” on page 9
- “Installation” on page 25
- “LABCAR-MODEL-VVTB” on page 28
- “LABCAR-MODEL-ICE” on page 61
- “LABCAR-MODEL-PMSM” on page 88
- “LABCAR-MODEL-FC” on page 137
- “Licensing” on page 161
- “Glossary” on page 163
- “References” on page 165

1.2 Classification of Safety Messages

The safety messages used here warn of dangers that can lead to personal injury or damage to property:

DANGER

indicates a hazardous situation with a high risk of death or serious injury if not avoided.

WARNING

indicates a hazardous situation of medium risk, which could result in death or serious injury if not avoided.

CAUTION

indicates a hazardous situation of low risk, which may result in minor or moderate injury if not avoided.

NOTICE

indicates a situation, which may result in damage to property if not avoided.

1.3 Presentation of Instructions

The target to be achieved is defined in the heading. The necessary steps for his are in a step-by-step guide:

Target definition

1. Step 1
2. Step 2
3. Step 3

> Result

1.4 Typographical Conventions

Software

`OCI_CANTxMessage msg0 =`	Code snippets are presented in the Courier font. Meaning and usage of each command are ex-plained 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 in small caps.
The "Open File" dialog box is displayed.	Names of program windows, dialog boxes, fields, etc. are shown in quotation marks.
Select the file `setup.exe`.	Text in drop-down lists on the screen, pro-gram code, as well as path- and file names are shown in the Courier font.
A *distribution* is always a one-dimensional table of sample points.	General emphasis and new terms are set in italics.

1.5 Presentation of Supporting Information

NOTE

Contains additional supporting information.

1.6 Privacy Notice

Your privacy is important to ETAS. We have created the following privacy notice that informs you, which data are processed in LABCAR-MODEL, which data categories LABCAR-MODEL uses, and which technical measure you have to take to ensure the privacy of the users. Additionally, we provide further instructions where this product stores and where you can delete personal or person-related data.

1.6.1 Data Processing

Note that personal or person-related data respectively data categories are processed when using this product. The purchaser of this product is responsible for the legal conformity of processing the data in accordance with Article 4 No. 7 of the General Data Protection Regulation (GDPR). As the manufacturer, ETAS GmbH is not liable for any mishandling of this data.

1.6.2 Data and Data Categories

When using the ETAS License Manager in combination with user-based licenses, particularly the following personal or person-related data respectively data categories can be recorded for the purposes of license management:

- Communication data: IP address
- User data: UserID, WindowsUserID

1.6.3 Technical and Organizational Measures

This product does not itself encrypt the personal or person-related data respectively data categories that it records. Please ensure the data security of the recorded data by suitable technical or organizational measures of your IT system, e.g., by classical anti-theft and access protection on the hardware.

Personal or person-related data in log files can be deleted by tools in the operating system.

2 LABCAR-MODEL Family

2.1 Overview of Models

The ETAS LABCAR solutions provide ways for testing and validating (TV) a wide range of automotive ECUs. The usage of numerical simulation models, which mimic the physical behavior of the System under test (SUT), is a widely known method. It offers many advantages over using real equipment as SUT, like reproducibility of results, parallel execution, access to all physical and auxiliary quantities, and the possibility of reconfiguration when a different SUT is needed. Frequently, such models do not resemble all the details of the SUT, but cover only the relevant aspects needed for a closed-loop operation. Thus, the simulation models are a vital part of the entire TV solution; without a model, no closed-loop operation is possible.

The ETAS LABCAR-MODEL product family encompasses several domain-specific simulation models and the virtual vehicle test bench. The former models mimic the functionality of a SUT for a single domain. Currently, the following models are available:

- ETAS LABCAR-MODEL-FC is a realtime-capable simulation model of a fuel-cell system including a PEM fuel-cell stack, a hydrogen supply, an oxygen supply, and a coolant supply. The stack model includes a 1-D membrane model for a detailed description of the membrane resistance, water content, and the resulting water exchange between the electrodes. ETAS LABCAR-MODEL-FC allows you to perform test and validation of an automotive fuel-cell control unit (FCCU) in closed-loop.
- ETAS LABCAR-MODEL-ICE is a generic model of a Diesel, gasoline, and CNG internal combustion engine (ICE) and its subsystems. It includes detailed models of the combustion system, the fuel system, the intake and exhaust system, and the aftertreatment system. This model allows you to do test and validation of an engine management system (EMS).
- ETAS LABCAR-MODEL-PMSM is an FPGA-based model for a permanent magnet synchronous electrical motor. It offers high performance and low latency computations. This model allows you to do test and validation of an e-motor control unit (MCU).
- ETAS LABCAR-MODEL-VVTB is the virtual vehicle test-bench (VVTB). It contains simplistic models for all the components that make up a virtual vehicle. It serves as the integration platform with generic interfaces for all other - typically more sophisticated - plant models. You can extend LABCAR-MODEL-VVTB by your own domain-specific simulation models or you can extend it by one of the domain-specific simulation models of the ETAS LABCAR-MODEL family.

2.2 General Model Parametrization

2.2.1 Obtaining the Data for Parametrization

The most important point for every parametrization is the determination of the correct data.

The first step in the model parametrization is collecting parametrization data. For this purpose, each LABCAR-MODEL product comes with a detailed model reference guide including a parameter list. You can query the availability of each parameter.

There are several sources of the data. You can take model parameters directly from an ECU. The prerequisite for this is a calibrated ECU software. It is essential that the ECU data used for the model parametrization is consistent with the data used in the ECU.

The second possibility involves the parameters for the parametrization of the open-loop model. This guarantees the correct generation and measurement of signals. This is the foundation of a LABCAR operation.

The actual model parameters are addressed in the third step. This part contains the most important parameters of the standard model variants.

The last step is for other tasks, which depend on the project in terms of the parametrization. Further sources of information for the parametrization are data sheets for special vehicle components, measurement protocols from the engine test bench, or other accessible data.

NOTE

If you have an existing LABCAR-OPERATOR project, you may reduce the amount of work needed for the parametrization by using your old parameters as a starting point. The release notes of LABCAR-MODEL list all changes in the meaning of the parameters between the different versions.

2.2.2 Model Parametrization Process

The model parametrization distinguishes between offline and online parametrization. The two ways are described in the following sections.

2.2.2.1 General Parametrization Process

LABCAR-MODEL products provide several predefined configurations, e.g., different engine configurations, which include plausible default values for parametrization. In the context of the parametrization process, these default data sets define the starting point for individual adaptations to suit the respective project. This takes place in two main steps, the offline and the online parametrization. Once the parametrization is completed, the model can be productively used to test a real or virtual ECU by modifying the parameters, using their interfaces. Furthermore, you can modify the model to execute error simulations. While the model is in use for test and validation, new demands for adjusting parameters arise or additional data for parametrization is supplied. In that case, the parametrization has to be re-applied following the same parametrization strategy as initially. This procedure is depicted in Fig. 2-1.

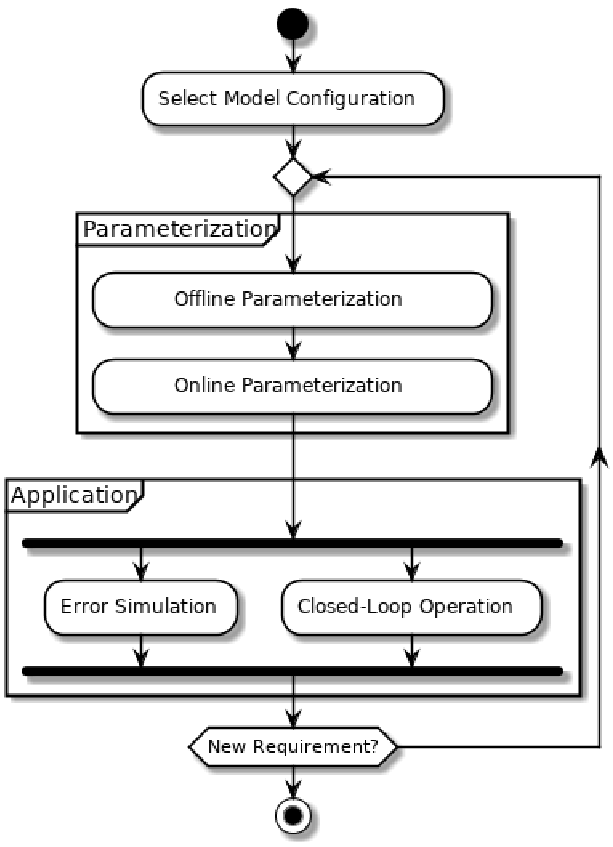


Fig. 2-1 General Parametrization Workflow

2.2.2.2 Offline Parametrization

Before the offline parametrization you select the best fitting initial model configuration for the particular project. With that, you define the fundamental vehicle topology, e.g., the amount of engine turbochargers fitting its engine topology.

The parametrization starts focusing on fundamental parameters. Here, you can eliminate existing deviations between initial model and desired vehicle model configuration, e.g., by setting the number of cylinders. In a second step, you set parameters which contribute to the precision of the model. In that stage of precise parametrization, you can set parameters which are responsible for the "fine tuning" of the model. Execute this part of the offline parametrization only if data is available for precise parametrization.

2.2.2.3 Online Parametrization

Once the offline parametrization is completed, you can continue with the online parametrization. The unit under test, either a physical or a virtual ECU, is now connected to the project. Ensure that the parameter file from the offline parametrization is available in LABCAR-EE and the simulation is running.

For a successful online parametrization do the following

1. Calibrate the ECU.
2. If used, configure and check the hardware boards. (Then signals are measured and sent correctly).

3. Correctly set the parameters in the open-loop model.

Like the offline parametrization, the online basic parametrization involves the parameters being modified in the same order, i.e., according to the model functionalities. See figure Fig. 2-2 on page 12. You only select insufficiently parametrized model functionalities iteratively and deactivate the model or the ECU functionality, which is not in focus. If available, all the switch parameters are checked. This could actually solve the problem of insufficient parametrization. If not, steady-state operating points are set in the model at which the simulation results are compared with target values, e.g., the engine test bench measurements. You adapt the relevant parameters until the model precision is sufficient and passes different steady state operation points. With finishing the parametrization, the process repeats for the next insufficiently parametrized model functionality until the overall model matches the expected behavior.

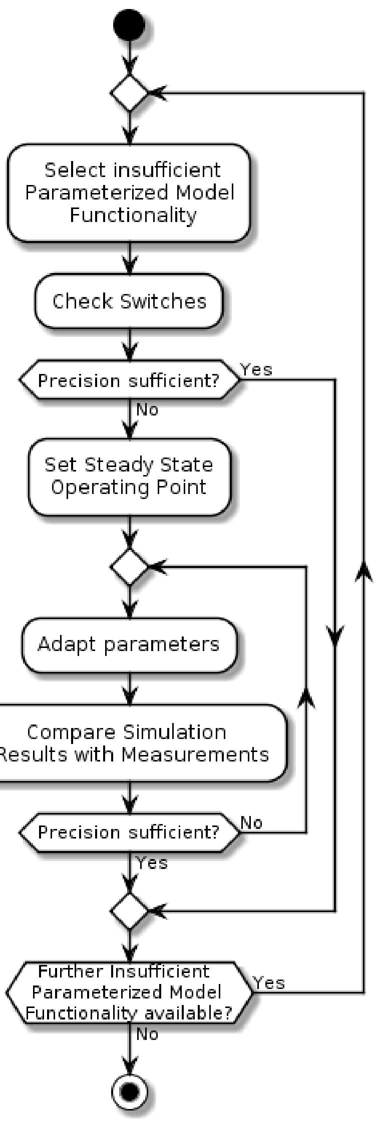

Select insufficient Parameterized Model Functionality
Check Switches
Precision sufficient?
Yes
No
Set Steady State Operating Point
Adapt parameters
Compare Simulation Results with Measurements
Precision sufficient?
No
Yes
Further Insufficient Parameterized Model Functionality available?
Yes
No

Fig. 2-2 Online Parametrization Workflow

2.3 Technical Aspects

This section comprises various technical points.

2.3.1 LABCAR Port Blocks

To enable communication with the LABCAR hardware as well as to connect to other simulation models in the LABCAR-IP, appropriate interfaces are necessary for Simulink modules. Instead of the Simulink inputs and outputs so-called LCInports and LCOutports are used for this purpose. After the installation of LABCAR-OPERATOR, these ports are available in the Simulink library. Simulink based LABCAR-MODEL products provide already various LCPorts for a faster integration. For further information about the Simulink integration in LABCAR-OPERATOR products, see [1].

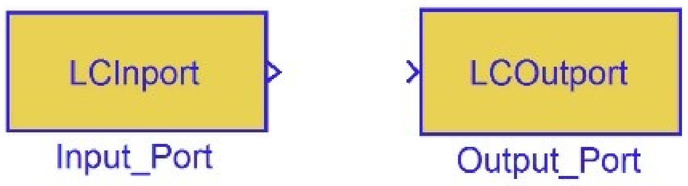


Fig. 2-3 LCInports and LCOutports

LCPorts are multifunctional interfaces, which can be of great benefit during the closed-loop deployment. They provide the following key features:

Access:
Each LCPort is accessible via the workspace of LABCAR-EE.

Logging:
Signal traces can be generated using the Datalogger functionality of LABCAR-EE.

Manipulation:
Manual signal manipulation is possible by overwriting the current signal value.

Stimulation:
Automatic signal manipulation is possible using the Signal Generator of LAB-CAR-EE by overwriting the current signal value.

	Simulink Inport	Simulink Outport	LCInport	LCOutport
Access	No	Yes	Yes	Yes
Logging	No	Yes	Yes	Yes
Manipulation	No	No	Yes	No
Stimulation	No	No	Yes	No

Tab. 2-1 Simulink Port and LCPort Functionality in LABCAR-OPERATOR and COSYM

2.3.2 Simulink Model - Code Generation Process

LABCAR-OPERATOR

Simulink based simulation models have to go through a code generation process when:

- You add a model to a project.
- You change the model `*.mdl` file.
- You explicitly request the code generation.

Since the code generation is crucial for the operation of LABCAR-MODEL, the following section briefly describes the technical fundamentals.

During the code generation, the Simulink `*.mdl` file is translated into ANSI C-code for a particular target, e.g. the ETAS RTPC. This translation takes place in two steps. First, the model `*.mdl` file is translated into a model description file `*.rtw`. The `*.rtw` file represents the original model in a high-level language and is input for the subsequent "Target Language Compiler". In a second step, C-code is specified for each Simulink block, based on a rule set described in a particular `*.tlc` file. As result, the "Target Language Compiler" generates target compatible C-code. Simulink based LABCAR-MODEL products officially support the code generation with the `lcrt.tlc` file that is shipped with LABCAR-OPERATOR and COSYM.

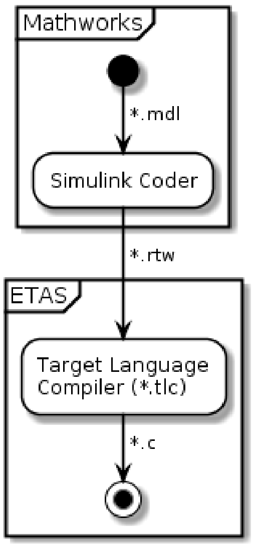


Fig. 2-4 Code Generation Process for Simulink Models

NOTE

The `grt.tlc` file is not officially supported by LABCAR-MODEL products.

NOTE

The generated code may differ depending on the MATLAB versions, although the same `*.tlc` file is used. In most of such cases it effects the representation of model parameters in LABCAR-EE.

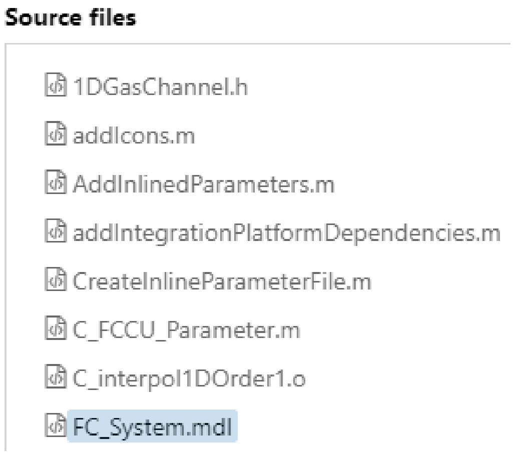
COSYM

Unlike LABCAR-OPERATOR where the Simulink code generation is re-triggered during every build, COSYM uses the already built sources of Simulink models. If any modifications are needed in a Simulink model, the corresponding changes need to be transferred to C-code using the following procedure.

To modify a Simulink model

1. Open the "Simulink model properties" view in COSYM with a double click on the Simulink model.

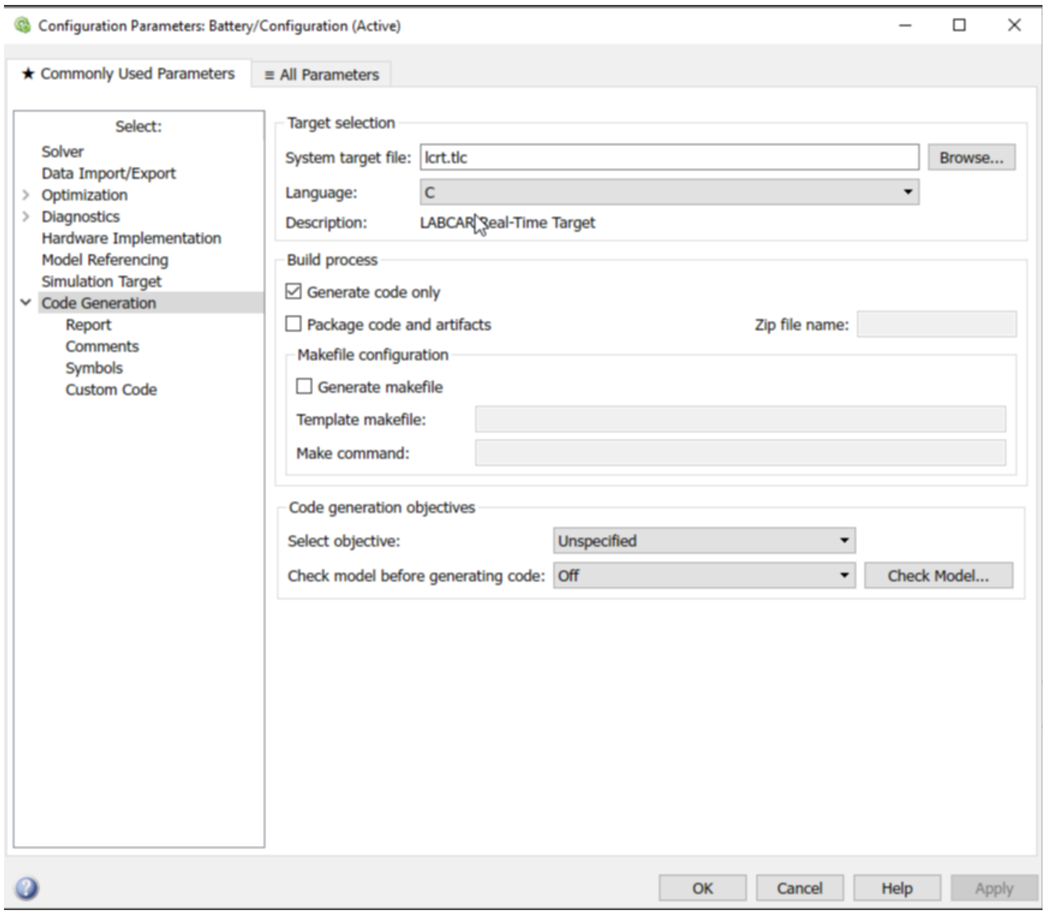
2. Browse to the `*.mdl` file and double click on it to open the Simulink model in the MATLAB®/Simulink® tool.



Source files

- 1DGasChannel.h
- addIcons.m
- AddInlinedParameters.m
- addIntegrationPlatformDependencies.m
- CreateInlineParameterFile.m
- C_FCCU_Parameter.m
- C_interpol1DOrder1.o
- FC_System.mdl

3. Make the necessary changes or updates to the model.
4. Perform the code generation by clicking on **Run** in the MATLAB/Simulink tool. Select the `lcrt.tlc` target in the configuration of the Simulink model.



Configuration Parameters: Battery/Configuration (Active)

★ Commonly Used Parameters | ≡ All Parameters

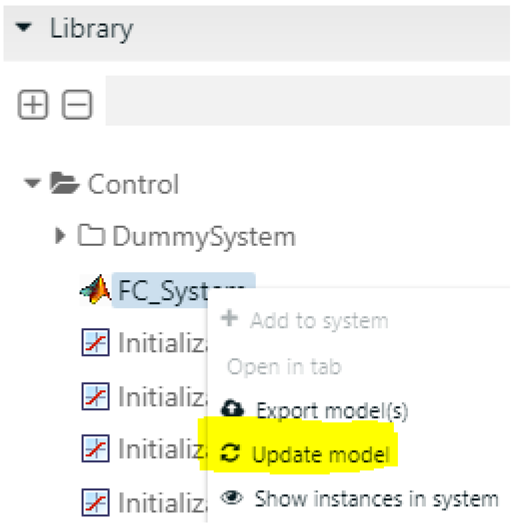
Select: Solver, Data Import/Export, Optimization, Diagnostics, Hardware Implementation, Model Referencing, Simulation Target, Code Generation (Report, Comments, Symbols, Custom Code)

Target selection
System target file: lcrt.tlc
Language: C
Description: LABCAR Real-Time Target

Build process
☑ Generate code only
☐ Package code and artifacts — Zip file name:
Makefile configuration
☐ Generate makefile
Template makefile:
Make command:

Code generation objectives
Select objective: Unspecified
Check model before generating code: Off

5. Now in COSYM, right click on the Simulink model and perform an update of the sources using the "Update model" pop-up menu item.

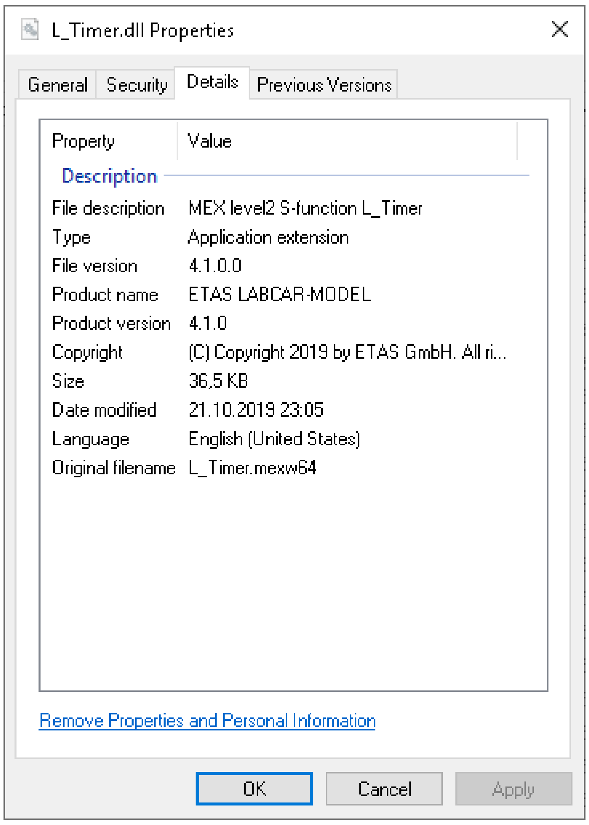


2.3.3 MEX File Version Information

The MATLAB S-Functions included in the LABCAR-MODEL products carry an embedded version information. For accessing the version information of a specific S-Function, its file extension has to be changed from `mexw64` to `dll`. Afterwards, it becomes accessible by right-clicking the file and opening the "Properties" window of this file. The version information (cf. Fig. 2-5) is listed in the "Details" tab of the properties dialog.

NOTE

Please consider this version information when sending an error report about a single S-Function.



Property	Value
Description	
File description	MEX level2 S-function L_Timer
Type	Application extension
File version	4.1.0.0
Product name	ETAS LABCAR-MODEL
Product version	4.1.0
Copyright	(C) Copyright 2019 by ETAS GmbH. All ri...
Size	36.5 KB
Date modified	21.10.2019 23:05
Language	English (United States)
Original filename	L_Timer.mexw64

Fig. 2-5 Exemplary Version Information for an S-Function

2.3.4 Mapping External Files for HiL Execution

All Simulink-based LABCAR-MODEL products use pre-compiled binaries for the execution. For running a model in Simulink on a Windows OS, according dynamic link libraries (`*.dll`) and Mex-Functions (`*.mexw64`) are available in each model folder. For the real-time execution on an RTPC, equivalent binaries are available as object files (`*.o`) or as static link libraries (`*.a`). The following section describes how to use these `*.a` and `*.o` files to build a COSYM or LABCAR-OPERATOR project for the HiL execution.

In addition, each Simulink-based model product comes with the following files for the HiL execution:

- `libChecker.a`

  Mandatory file for licensing
- `libLicenseChecker.a`

  Mandatory file for licensing
- `<Mex-Function>.o`

  Mex-Function object code for each S-Function used by the model

All given `*.a` and `*.o` files in the model folder must be added as external dependencies to the COSYM or LABCAR-OPERATOR project. COSYM and LAB-CAR-OPERATOR offer the option to reference additional sources or binaries for a model by using the "External Files" option. A detailed step-by-step guide is given in section "Model Management", in the COSYM User Guide [6]. Ensure that all required `*.o` files for the model and all `*.a` files for the licensing are added.

NOTE

The project items folder contains already pre-configured Simulink models. They include a file `ExternalFiles.xml`, which is parsed by COSYM or LABCAR-OPERATOR as reference file when integrating the model directly from the project items folder. All required external files in the `ExternalFiles.xml` file are automatically set. This is also used in section 4.1.1, 5.1.1 and 7.1.1 for the corresponding creation of example projects.

2.4 LABCAR-MODEL Based FMUs

Starting with V4.3, the LABCAR-MODELs include a workflow to export LCRT-compatible Simulink models as Function Mock-up Units (FMUs). In this way, they can be used in COSYM-SiL or any other co-simulation tool, which supports FMI-based model integration.

For keeping the compatibility with LABCAR-OPERATOR, the models continue to use LC-Inports and LC-Outports. These ports serve as interface to the simulation in the RTPC via LABCAR-EE. They need to be replaced by standard Simulink ports at the root level, before the FMU generation can take place. To this end, an FMU exporter tool is introduced which facilitates the conversion of Simulink based LABCAR-MODELs to FMI 2.0 compatible FMUs.

The following table shows the LABCAR-MODEL products, the FMI version, and the supported targets:

LABCAR-MODEL Product	FMI 2.0 for Co-Simulation	FMI 2.0 for Model Exchange
LABCAR-MODEL-ICE	Win64	-
LABCAR-MODEL-VVTB	Win64	-
LABCAR-MODEL-FC	Win64	-
LABCAR-MODEL_PMSM	-	-

2.4.1 FMU Exporter Tool

FMU Exporter is a MATLAB based command line interface tool that provides an API to migrate LCRT-compatible Simulink models to FMUs. The tool abstracts the functionality provided by FMI-Kit for the generation of FMUs.

The following phases are involved in translating a Simulink model to FMU:

- To identify and replace LC-Ports with Standard Simulink ports at root level
- To locate the S-function dependencies for the models, which are needed for the code generation
- To perform the code generation using the GRT-FMI target file of FMI-Kit.

The FMU Exporter tool is installed in a common location along with product installers:

`C:\Program Files\Common Files\ETAS\LABCAR-MODEL\4.3\FMUExporter\`

> **NOTE**
>
> Before using this FMU Exporter tool, install FMI-Kit (section 2.4.2 "FMIKit-Simulink Library" on page 23).

The following methods are provided by the FMU exporter tool:

- LABCARMODEL.ExportAsFmu()
- LABCARMODEL.ExportAsFmuWithConfig()
- LABCARMODEL.CreateExportConfigurationTemplate()
- LABCARMODEL.ReplaceLabcarPorts()

ExportAsFMU

`LABCARMODEL.ExportAsFmu(sourceModelFilePath, targetFmuFilePath)`

The function `ExportAsFmu` creates an FMU from a LABCAR-MODEL Simulink model. The specified model is exported with default code generation settings.

The function `ExportAsFmu` requires two input arguments as the following parameters.

- **`sourceModelFilePath (char array(1,:)) -`**

  is the file path to the existing `*.mdl` model file, which shall be exported. Both absolute paths and relative paths are accepted, as well as plain file names, which can be found on the MATLAB path.

- **`TargetFmuFilePath (char array(1,:))`**

  is the file path to which the generated `*.fmu` model file shall be saved. A folder path is not sufficient, a file name is expected as part of the path. Both absolute paths and relative paths are accepted, as well as plain file names, which are relative to the current working folder. If it already exists, it will be overwritten.

Find some examples in the following:

- `LABCARMODEL.ExportAsFmu('VVTB.mdl', 'VVTB.fmu')`
- `LABCARMODEL.ExportAsFmu('C:\Program Files\ETAS\LABCAR-MODEL\V4.2.0\VVTB\Simulink Models\VVTB_MT\VVTB.mdl', 'C:\Temp\MyVVTBFMU.fmu')`

<u>To show the help text for this function in MATLAB</u>

1. Type the command `help LABCARMODEL.ExportAsFmu`.

   The help information appears directly in the MATLAB command line window.

or

1. Type the command `doc LABCARMODEL.ExportAsFmu`.

   The help information appears in a separate HTML page.

ExportAsFmuWithConfig

`LABCARMODEL.ExportAsFmuWithConfig(sourceModelFilePath, targetFmuFilePath, configMFilePath)`

The function `ExportAsFmuWithConfig` creates an FMU from a LABCAR-MODEL Simulink model with custom code generation settings. These are applied from a previously generated configuration file.

The function `ExportAsFmuWithConfig` requires three input arguments as the following parameters:

- **`sourceModelFilePath (char array(1,:))`**

  is the file path to the existing `*.mdl` model file, which shall be exported. Both absolute paths and relative paths are accepted, as well as plain file names that can be found on the MATLAB path.
- **`targetFmuFilePath (char array(1,:))`**

  is the file path to which the generated `*.fmu` model file shall be saved. A folder path is not sufficient, a file name is expected as part of the path. Both absolute paths and relative paths are accepted, as well as plain file names, which are relative to the current working folder. If it already exists, it will be overwritten.
- **`configMFilePath (char array(1,:))`**

  is the file path to the existing export configuration `*.m` file, which shall be applied. A matching configuration file can be created using the function `LABCARMODEL.CreateExportConfigurationTemplate`. Both absolute paths and relative paths are accepted, as well as plain file names that can be found on the MATLAB path.

Find some examples in the following:

```
– LABCARMODEL.ExportAsFmuWithConfig('VVTB.mdl',
  'VVTB.fmu', 'MyVVTBExportConfig.m')
– LABCARMODEL.ExportAsFmuWithConfig('C:\Program
  Files\ETAS\LABCAR-MODEL\V4.2.0\VVTB\Simulink
  Models\VVTB_MT\VVTB.mdl',
  'C:\Temp\MyVVTBFMU.fmu',
  'C:\Temp\MyVVTBExportConfig.m')
```

<u>To show the help text for this function in MATLAB</u>

1. Type the command
   `help LABCARMODEL.ExportAsFmuWithConfig`.

   The help information appears directly in the MATLAB command line window.

or

1. Type the command
   `doc LABCARMODEL.ExportAsFmuWithConfig`.

   The help information appears in a separate HTML page.

CreateExportConfigurationTemplate

`LABCARMODEL.CreateExportConfigurationTemplate(sourceModelFilePath, targetConfigMFilePath)`

The function `CreateExportConfigurationTemplate` creates a configuration file, which can be modified by the user and which can subsequently be used for the function `ExportAsFmuWithConfig` to generate an FMU with custom code generation options.

The function `CreateExportConfigurationTemplate` requires two input arguments as the following parameters:

- **`sourceModelFilePath (char array(1,:))`**

  is the file path to the existing `*.mdl` file, which shall be exported subsequently. Both absolute paths and relative paths are accepted, as well as plain file names that can be found on the MATLAB path.
- **`targetConfigMFilePath (char array(1,:))`**

  is the file path to which the generated `*.m` configuration template shall be saved. A folder path is not sufficient, a file name is expected as part of the path. Both absolute paths and relative paths are accepted, as well as plain file names which are relative to the current working folder. If it already exists, it will be overwritten.

Find some examples in the following:

- `LABCARMODEL.CreateExportConfigurationTemplate('VVTB.mdl', 'MyVVTBExportConfig.m')`
- `LABCARMODEL.CreateExportConfigurationTemplate('C:\Program Files\ETAS\LABCAR-MODEL\V4.2.0\VVTB\Simulink Models\VVTB_MT\VVTB.mdl', 'C:\Temp\MyVVTBExportConfig.m')`

<u>To show the help text for this function in MATLAB</u>

1. Type the command
   `help LABCARMODEL.CreateExportConfigurationTemplate`.

   The help information appears directly in the MATLAB command line window.

or

1. Type the command
   `doc LABCARMODEL.CreateExportConfigurationTemplate`.

   The help information appears in a separate HTML page.

ReplaceLabcarPorts

`LABCARMODEL.ReplaceLabcarPorts(sourceModelFilePath, targetModelFolderPath)`

The function `ReplaceLabcarPorts` removes LABCAR-HiL dependencies (LC-Inports, LC-Outports) from a LABCAR-MODEL Simulink model and replaces them with From\Goto blocks.

The function `ReplaceLabcarPorts` requires two input arguments as the following parameters:

- **`sourceModelFilePath (char array(1,:))`**

  is the file path to the existing `*.mdl` model file, which shall be exported. Both absolute paths and relative paths are accepted, as well as plain file names that can be found on the MATLAB path.
- **`targetModelFolderPath (char array(1,:))`**

  is the folder path to which the modified model shall be saved. The folder does not need to exist, but shall be empty if it does. Existing files in the folder may be overwritten. Both absolute paths and relative paths are accepted.

Find some examples in the following:

- `LABCARMODEL.ReplaceLabcarPorts('VVTB.mdl', 'MyModifiedModelFolder')`
- `LABCARMODEL.ReplaceLabcarPorts('C:\Program Files\ETAS\LABCAR-MODEL\V4.2.0\VVTB\Simulink Models\VVTB_MT\VVTB.mdl', 'C:\MyModifiedModelFolder')`

<u>To show the help text for this function in MATLAB</u>

1. Type the command `help LABCARMODEL.ReplaceLabcarPorts`.

   The help information appears directly in the MATLAB command line window.

or

1. Type the command `doc LABCARMODEL.ReplaceLabcarPorts`.

   The help information appears in a separate HTML page.

2.4.2 FMIKit-Simulink Library

FMIKit-Simulink is an open source Simulink library to import and export Function Mock-up Units (FMU) based on FMI V1.0 and V2.0 for Model Exchange and Co-Simulation.

FMU Exporter uses this library for exporting the LABCAR-MODELs as FMUs. To use the FMU Exporter tool, the latest version of FMIKit-Simulink library V2.9 is needed.

For additional information and the download of the FMIKit-Simulink library, please refer to the official "GitHub" page
https://github.com/CATIA-Systems/FMIKit-Simulink .

To use the FMU Exporter

1. Download FMIKit 2.9 from the FMIKit download link.
2. Extract it to a folder of your choice.
3. Refer to FMIKit Quick start to complete the FMI-Kit setup in MATLAB.

3 Installation

This section contains information on the scope of delivery and on how to prepare for the installation.

3.1 User Privileges Required for Installation

You must have administrator privileges to install products of LABCAR-MODEL on a PC. If in doubt, ask your system administrator.

3.2 Contents of the DVD

The products of LABCAR-MODEL are delivered as a DVD with the following contents:

- ETAS_Safety_Advice.pdf
- Release_Notes.pdf
- Documentation:

  Contains the User's Guide for LABCAR-MODEL products and Open Source Scan (OSS) attribution documents.
- Installation:

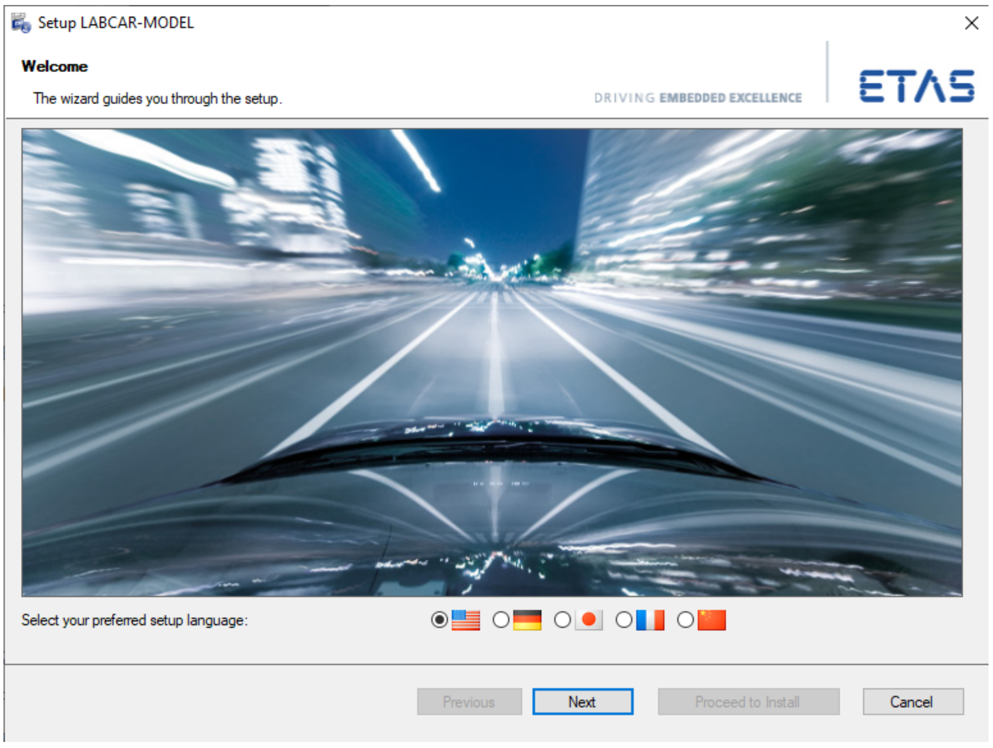
  Contains installer executable including all required installation packages.

3.3 Program Installation

This section describes how to install LABCAR-MODEL products.

1. Insert the product DVD in your DVD-ROM drive.

   The DVD-Browser is launched.
2. Run Installation/setup.exe to start the installer manually.


Setup LABCAR-MODEL
Welcome
The wizard guides you through the setup.
DRIVING EMBEDDED EXCELLENCE
ETAS
Select your preferred setup language:
Previous
Next
Proceed to Install
Cancel

3. Click **Next** and follow the instructions of the installation wizard.
4. To complete the installation click **Finish**.

3.4 Contents of the Installation

The contents of the installation consists of the following:

- Simulink Models

  There are Simulink models including compiled S-Functions in the form of `*.mexw64` files for Microsoft Windows and `*.o` files for ETAS RTPC. For the integration into LABCAR-OPERATOR or COSYM projects, use the Simulink models bundled as project items instead of the pure Simulink models in this area of the installation. You find the description in the section “Project Items” on page 26.
- FPGA Models

  There are FPGA programming file (`*.euf)` and interface description files (`*.adf`) for flashing the ES5340.2.
- Project Items

  Project items are bundles of supplementary items for creating projects in LABCAR-OPERATOR or COSYM. The items are tailored to the aforementioned products for simple integration into new or existing LABCAR projects.

  Items in the subfolder `LCO` are compatible to LABCAR-OPERATOR, items in the subfolder `COSYM` are compatible to COSYM.

  Some items are specific to individual model variants as indicated by subfolder names. Inside these variant subfolders, individual items are again categorized.

  The items comprise the following:

  – Simulink models

    You find Simulink models inside the `mdl` subfolders, which are prepared for direct import to LABCAR-OPERATOR or COSYM. The same model variants are available here as described in the section “Simulink Models” on page 26.

  – C-modules

    You find C-modules inside the `cmod` subfolders, which can be helper functions for initializing ambient conditions, for example.

  – Connection files

    You find connection files inside the `conn` subfolders, which help in connecting different models, e.g. LABCAR-MODEL-VVTB and LABCAR-MODEL-ICE, and C-modules to each other.

  – Layers

    You find layers inside the `gui` subfolders. They can be added to an experiment in LABCAR-EE to provide an intuitive user interface with handy controls for commonly used parameters and interesting observables.

  – Images

    You find images inside the `img` subfolders. They are required for the supplied experiment layers, which are located inside the `gui` subfolders, to work as expected.

- Mapping files

  You find the mapping files inside the `mapping` subfolders. They provide abstractions for signals inside a model and are required for experiment layers, which are located inside the `gui` subfolders, to work as expected.
- RT-plugins

  You find the RT-plugins inside the `rtp` subfolders. They provide additional functionality required for experiment layers, which are located inside the `gui` subfolders, to work as expected.
- Specific user-data

  You find the specific user data inside the `userdata` subfolders.

When creating a project, choose the items from the subfolder for the desired model variant. Project items for different models variants differ from each other and are not interchangeable. For example, connection files found in the project items of LABCAR-MODEL-VVTB differ between the automatic transmission model variant (VVTB_AT) and the manual transmission model variant (VVTB_MT).

You can find more information on how to use the specific project items delivered with LABCAR-MODEL products in the product-specific sections of this User's Guide.

For general information about items like layers, mapping files, connection files, etc., and how to work with them please refer to [1].

- LicenseConfigurationInfo:

  These are files for the ETAS License Manager.
- Example Project:

  Pre-built example projects for LABCAR-OPERATOR are available in the `LCO` subfolder of this area. Example projects for COSYM may be found in the `COSYM` subfolder for products supporting it.
- Documentation:

  This area contains the User's Guide, Safety Advices as well as individual Model Reference Guides.

NOTE

Please do not open or manipulate the files in the installation folder directly. Instead create a working copy from the installation folder in a location of your choice.

4 LABCAR-MODEL-VVTB

ETAS LABCAR-MODEL-VVTB is the virtual vehicle test bench. On the one hand it serves as the top-level vehicle model and allows for an easy integration of all those sophisticated, domain-specific component models that you need to describe the full SUT. On the other hand it provides simplistic default implementations for all subsystems, where no sophisticated component models are needed.

4.1 Project and Experiment Creation

In the following, step-by-step guides describe the generation of a COSYM project or a LABCAR-OPERATOR project using LABCAR-MODEL-VVTB. For the execution of the project, no physical ECU is needed. Instead, a SoftECU controls the engine model. The project to be created contains a manual transmission vehicle model that is predefined by the VVTB_MT model configuration.

4.1.1 Project Creation in COSYM

This section addresses qualified personnel with knowledge in using COSYM. If you do not have any experience in using COSYM, please read first [6].

For the manual project creation using LABCAR-MODEL-VVTB with COSYM follow the steps described in this section.

To set up the COSYM project

1. Open the COSYM application.

   The COSYM application can already have a open project. In this case

2. Click **File** → **New** → **Project.**

or

   If there is no previous project loaded,

   - Click **Create new project**.

3. Choose the location for the project.and name the project.
4. Click **Create.**

NOTE

Refer to section 7.1 in [6] for additional help regarding the project creation in COSYM.

> The "Library" section of the project is empty to start with. A default system is created with the name "System" and automatically mapped to "HiL_Target".

To add a hardware module to the system

1. Double click on the pane "System".
2. Expand "System"
3. Right click on "HiL_Target" and select "Create RTIO module" from the drop-down menu.

> The "System" shows the added hardware module in "Diagram" view.

The full vehicle model consists of a single Simulink module. This module also contains driver, environment, vehicle, and SoftECU, which mimics the behavior of an EMS.

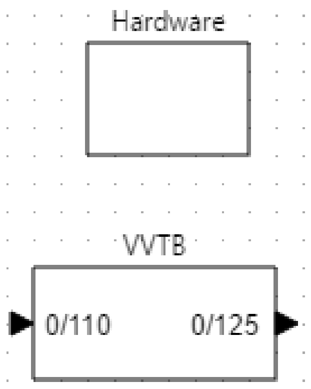
To add a Simulink module

1. In the "Library" pane, right click on the "Control" folder and choose "Import Model".
2. Under "Source of the model to be imported", click on **Browse** button.
3. Select the LMD file "`VVTB.lmd`", which internally refers to model configuration `VVTB.mdl.` The corresponding LMD file "`VVTB.lmd`" is located in the installation folder:

   `.\VVTB\Project Items\COSYM\VVTB_MT\mdl\Simulink_VVTB\src\VVTB_LABCAR_rtw.`
4. Click on the **Import** button.

   The model "VVTB" is added to the "Library" under the "Control" folder.
5. Drag and drop the "VVTB" model from the "Library" pane
   - to the "Diagram" area of the '"System"
   - and to the "Units" folder of "System" under the "Systems" pane.

> The "System" shows the added module in the "Diagram" view.



NOTE

The Simulink model references additional dependencies by using the `ExternalFiles.xml`. Please see section “Mapping External Files for HiL Execution” on page 18 for further details.

By default, the SoftECU is not connected to the vehicle or driver model, although it is part of the already integrated Simulink module `VVTB.mdl`. Pre-defined connection files couple the SoftECU and the vehicle model internally.

The following connection files have to be added:

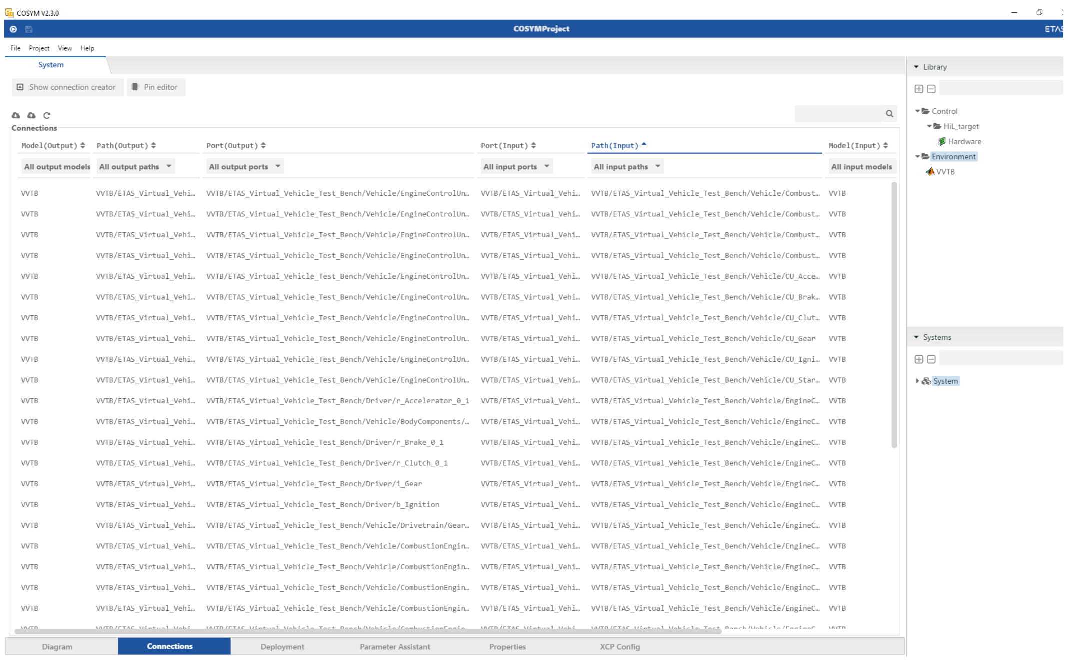
- `BodyComponents_Basic-CU_EMS.xml`
- `Combustion_Gasoline-CU_EMS.xml`
- `CU_EMS-Combustion_Gasoline.xml`
- `CU_EMS-Vehicle.xml`
- `CU_Testbed-VehicleDynamics_Longitudinal.xml`
- `Driver_Longitudinal-CU_Testbed.xml`
- `Driver-CU_EMS.xml`
- `Driver-CU_TCU_MT.xml`
- `Drivetrain_Longitudinal-CU_EMS.xml`
- `Drivetrain_Longitudinal-CU_TCU_MT.xml`
- `Drivetrain_Longitudinal-CU_Testbed.xml`

  These files are located in the installation folder at `.\VVTB\Project Items\COSYM\VVTB_MT\conn`.

To add connection files

1. Open the "Connections" view available at the bottom of the opened "System".
2. Click on the "Import connections" icon.
3. Select each XML file mentioned above.
4. Click on **Import** to import connections from each file.
5. Repeat the steps 1 to 4 for all the connection files mentioned above.

> After the connection files are added, the necessary connections are established between the VVTB and SoftECU. They are visible in the "Connections" view.



Model(Output)	Path(Output)	Port(Output)	Port(Input)	Path(Input)	Model(Input)
All output models	All output paths	All output ports	All input ports	All input paths	All input models
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineControlUn…	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/Combust…	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineControlUn…	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/Combust…	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineControlUn…	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/Combust…	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineControlUn…	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/Combust…	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineControlUn…	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/CU_Acce…	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineControlUn…	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/CU_Brak…	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineControlUn…	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/CU_Clut…	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineControlUn…	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/CU_Gear	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineControlUn…	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/CU_Igni…	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineControlUn…	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/CU_Star…	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Driver/r_Accelerator_0_1	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineC…	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/BodyComponents/…	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineC…	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Driver/r_Brake_0_1	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineC…	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Driver/r_Clutch_0_1	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineC…	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Driver/i_Gear	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineC…	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Driver/b_Ignition	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineC…	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/Drivetrain/Gear…	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineC…	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/CombustionEngin…	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineC…	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/CombustionEngin…	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineC…	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/CombustionEngin…	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineC…	VVTB
VVTB	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/CombustionEngin…	VVTB/ETAS_Virtual_Vehi…	VVTB/ETAS_Virtual_Vehicle_Test_Bench/Vehicle/EngineC…	VVTB

Fig. 4-1 COSYM Connections View for VVTB

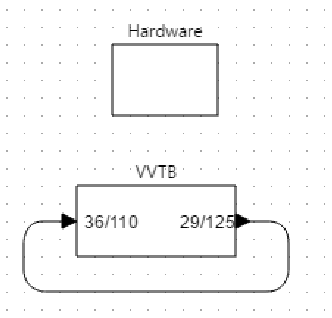


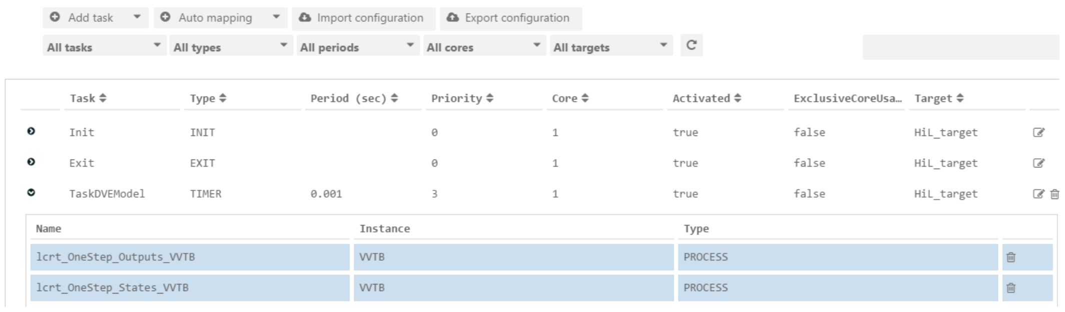
Fig. 4-2 COSYM Diagram View with Connections for VVTB

NOTE

Check each connection file and ensure that you added all connection files. Otherwise, the COSYM project might not work properly.

OS Configuration

Via the "Deployment" view, you can organize the processes of COSYM modules. For Simulink modules, the order of execution and the selection of a proper step size is important. By default, tasks are *not* auto-generated for Simulink modules when adding them to the project. Please click on **+ Auto mapping** to automatically map the processes to tasks.



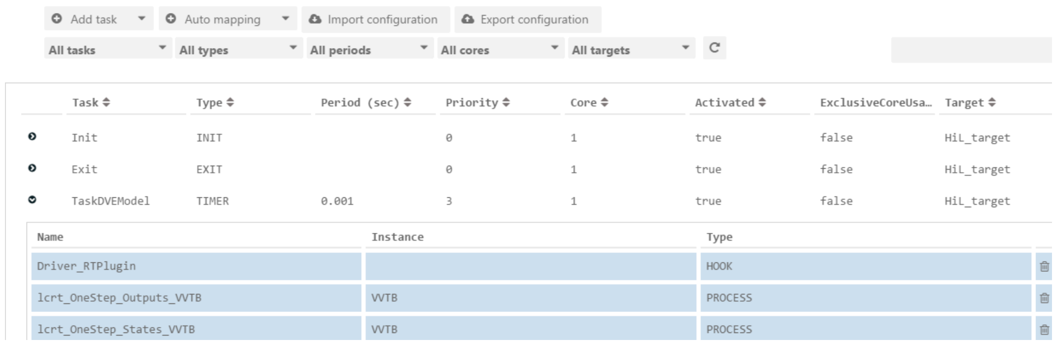
Task	Type	Period (sec)	Priority	Core	Activated	ExclusiveCoreUsa…	Target
Init	INIT		0	1	true	false	HiL_target
Exit	EXIT		0	1	true	false	HiL_target
TaskDVEModel	TIMER	0.001	3	1	true	false	HiL_target

Name	Instance	Type
lcrt_OneStep_Outputs_VVTB	VVTB	PROCESS
lcrt_OneStep_States_VVTB	VVTB	PROCESS

In addition to the model tasks, it is important to generate a process hook. The RT-Plugins require such process hooks.

To add RT-Plugins:

1. Right-click on `TaskDVEModel`.
2. Choose **Add Hook**.
3. Name the hook `Driver_RTPlugin`.
4. Move this hook before the process `lcrt_OneStep_Outputs_VVTB`.



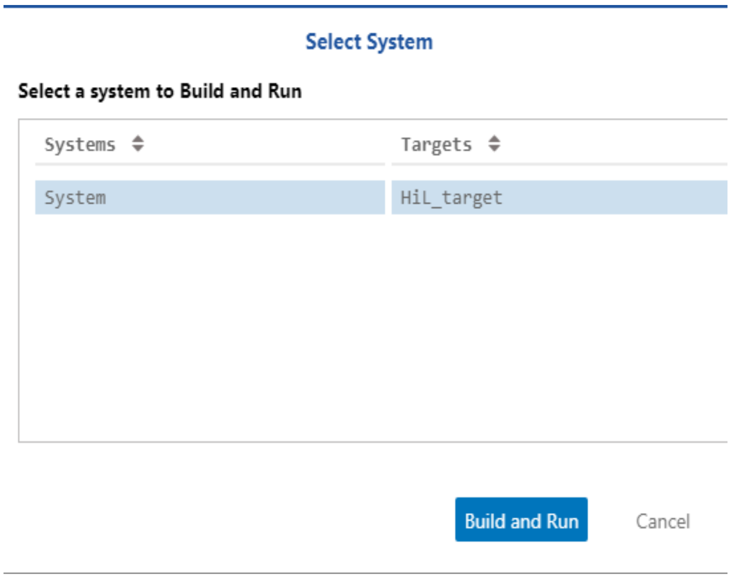
Task	Type	Period (sec)	Priority	Core	Activated	ExclusiveCoreUsa…	Target
Init	INIT		0	1	true	false	HiL_target
Exit	EXIT		0	1	true	false	HiL_target
TaskDVEModel	TIMER	0.001	3	1	true	false	HiL_target

Name	Instance	Type
Driver_RTPlugin		HOOK
lcrt_OneStep_Outputs_VVTB	VVTB	PROCESS
lcrt_OneStep_States_VVTB	VVTB	PROCESS

All relevant modules to operate the vehicle and the engine model are available and you can build a COSYM project.

To build a COSYM project

1. Choose **Project** → **Build and Run**.
2. Click **Build and Run.**



Select System

Select a system to Build and Run

Systems	Targets
System	HiL_target

For more details about the Simulink code generation, refer to “Simulink Model - Code Generation Process” on page 14.

4.1.2 Project Creation in LABCAR-OPERATOR

This section addresses qualified personnel with knowledge in using LABCAR-OPERATOR. If you do not have any experience in using LABCAR-OPERATOR, please read first [1].

For the manual project creation using LABCAR-MODEL-VVTB with LABCAR-OPERATOR follow the steps described in this section.

To set up the LABCAR-OPERATOR project

1. Open the LABCAR-IP application.
2. Click **File** → **New Project**.
3. Choose the location for the file and name the project.
4. Use "RTPC" as "Target Name".
5. Click **Finish**.

> The project explorer contains an empty hardware configuration.

The full vehicle model consists of a single Simulink module. This module also contains driver, environment, vehicle, and SoftECU, which mimics the behavior of an EMS.

To add a Simulink module

1. Choose **Project** → **Add Module.**
2. Select "Add Simulink (TM) Module".
3. Select "Use existing Simulink (TM) Model (will be copied)".

4. Select the model configuration **VVTB_MT**. The corresponding model `VVTB.mdl` is located in the install folder .\VVTB\Project Items\LCO\VVTB_MT\mdl\ Simulink_VVTB\src.
5. Activate the checkbox "Copy Model Directory into Project".
6. Click **Finish**.

> The Project Explorer shows the added module..

NOTE

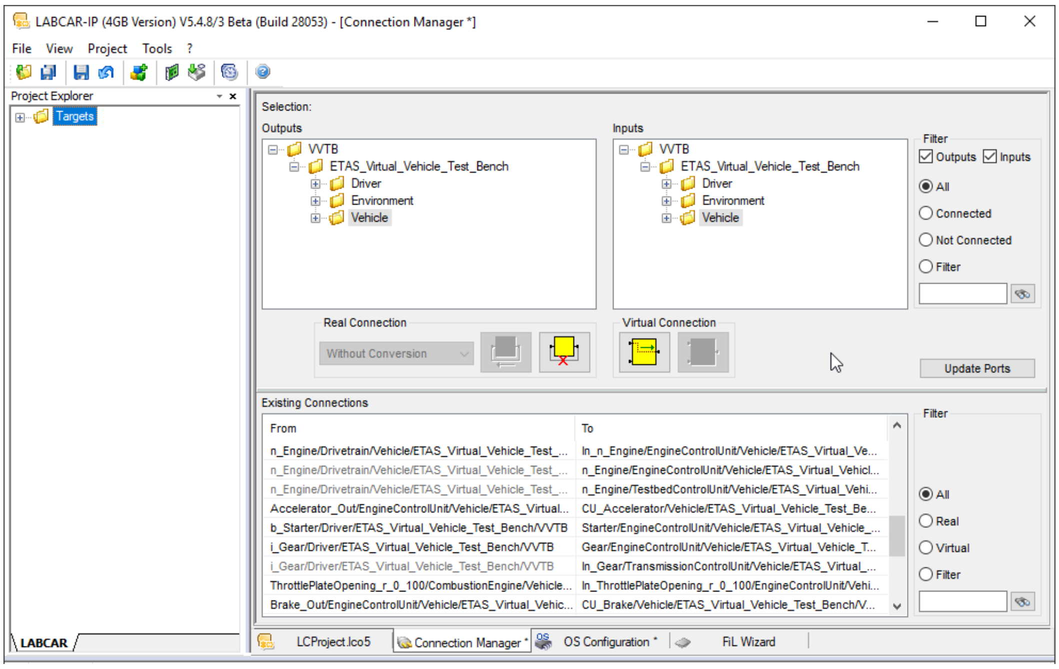
The Simulink model references additional dependencies by using the "`ExternalFiles.xml`". Please see section “Mapping External Files for HiL Execution” on page 18 for further details.

Per default, the SoftECU is not connected to the vehicle or driver model, although it is part of the already integrated Simulink module `VVTB.mdl`. Pre-defined connection files couple the SoftECU and the vehicle model internally.

To add connection files

1. Open "Connection Manager" and right-click.
2. Select **Import Connection**.
3. Import each connection file stored in the install folder at `.\VVTB\Project Items\LCO\VVTB_MT\conn.`

> After the connection files are added, many connections are visible in the "Connection Manager".



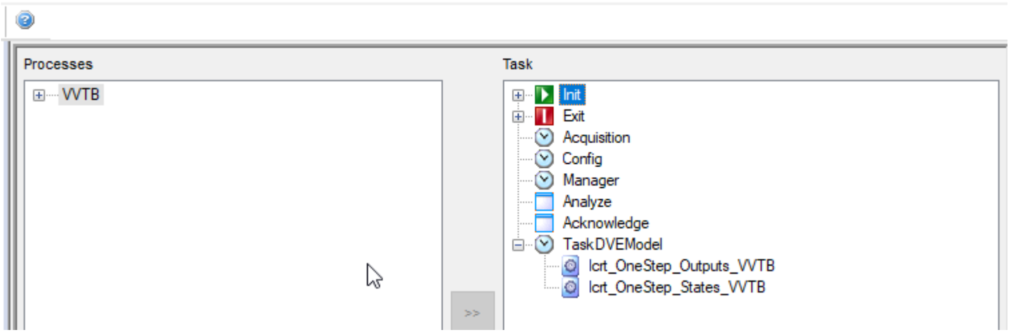
From	To
n_Engine/Drivetrain/Vehicle/ETAS_Virtual_Vehicle_Test_...	In_n_Engine/EngineControlUnit/Vehicle/ETAS_Virtual_Ve...
n_Engine/Drivetrain/Vehicle/ETAS_Virtual_Vehicle_Test_...	n_Engine/EngineControlUnit/Vehicle/ETAS_Virtual_Vehicl...
n_Engine/Drivetrain/Vehicle/ETAS_Virtual_Vehicle_Test_...	n_Engine/TestbedControlUnit/Vehicle/ETAS_Virtual_Vehi...
Accelerator_Out/EngineControlUnit/Vehicle/ETAS_Virtual...	CU_Accelerator/Vehicle/ETAS_Virtual_Vehicle_Test_Be...
b_Starter/Driver/ETAS_Virtual_Vehicle_Test_Bench/VVTB	Starter/EngineControlUnit/Vehicle/ETAS_Virtual_Vehicle_...
i_Gear/Driver/ETAS_Virtual_Vehicle_Test_Bench/VVTB	Gear/EngineControlUnit/Vehicle/ETAS_Virtual_Vehicle_T...
i_Gear/Driver/ETAS_Virtual_Vehicle_Test_Bench/VVTB	In_Gear/TransmissionControlUnit/Vehicle/ETAS_Virtual_...
ThrottlePlateOpening_r_0_100/CombustionEngine/Vehicle...	In_ThrottlePlateOpening_r_0_100/EngineControlUnit/Vehi...
Brake_Out/EngineControlUnit/Vehicle/ETAS_Virtual_Vehic...	CU_Brake/Vehicle/ETAS_Virtual_Vehicle_Test_Bench/V...

NOTE

Check each connection file and ensure that you added all connection files. Otherwise the model might not work properly.

OS Configuration

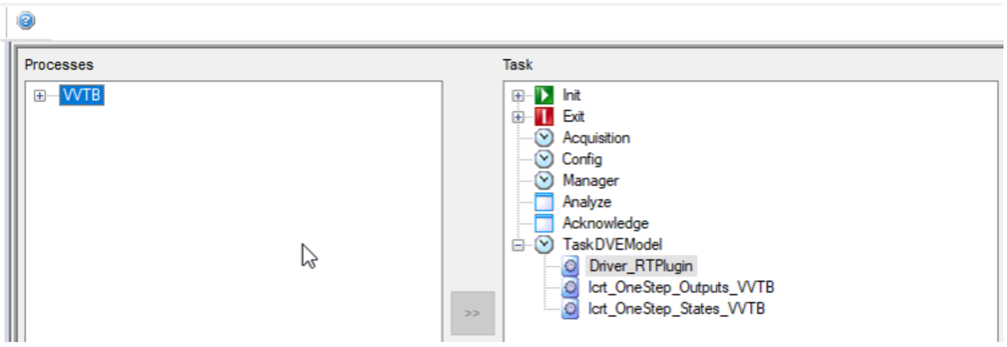
Via the "OS Configuration" tab, you can organize the processes of LABCAR-OPERATOR modules. For Simulink modules, the order of execution and the selection of a proper step size is of importance. Per default, proper tasks for the Simulink model are auto-generated when adding the module to the project.



In addition to the model tasks, it is important to generate a process hook. The RT-Plugins require such process hooks.

To add RT-Plugins

1. Right-click on `TaskDVEModel`.
2. Choose "`Add Hook`".
3. Name the hook `Driver_RTPlugin`.



> All relevant modules to operate the manual transmission vehicle model VVTB_MT are available.

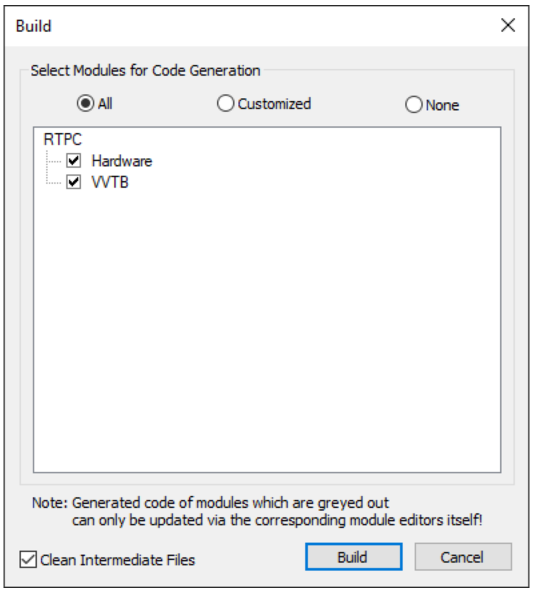
As next step, you can build the LABCAR-OPERATOR project.

NOTE

Be sure to select the proper MATLAB version using the MATLAB Version Selector. See [2].

To build a LABCAR-OPERATOR project

1. Choose **Project** → **Build.**
2. Activate the check box for the VVTB module.
3. Click **Build**.



Note: Generated code of modules which are greyed out can only be updated via the corresponding module editors itself!

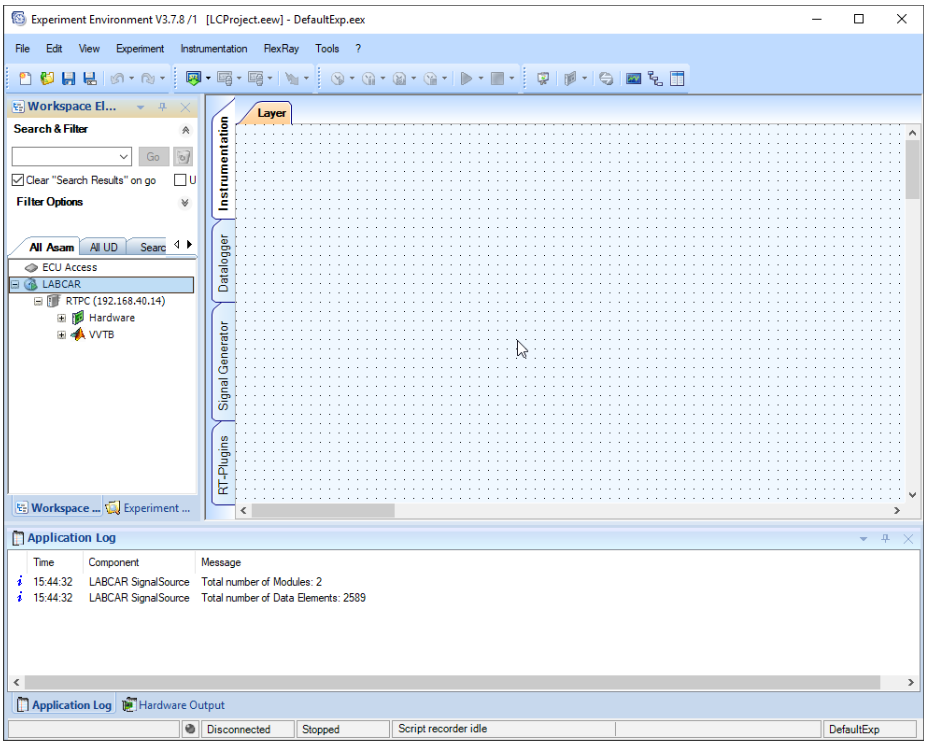
The build process performs the following basic steps:

- Simulink code generation on the host PC
- Transfer of the source code to RTPC
- Compilation of sources on RTPC
- Transfer of binaries back to the host PC

For more details about Simulink Code Generation, see “Simulink Model - Code Generation Process” on page 14.

4.1.3 Experiment Creation

After the successful build of the project in COSYM or LABCAR-OPERATOR, you can prepare the experiment for the first execution of the simulation model. Starting point is an empty experiment.

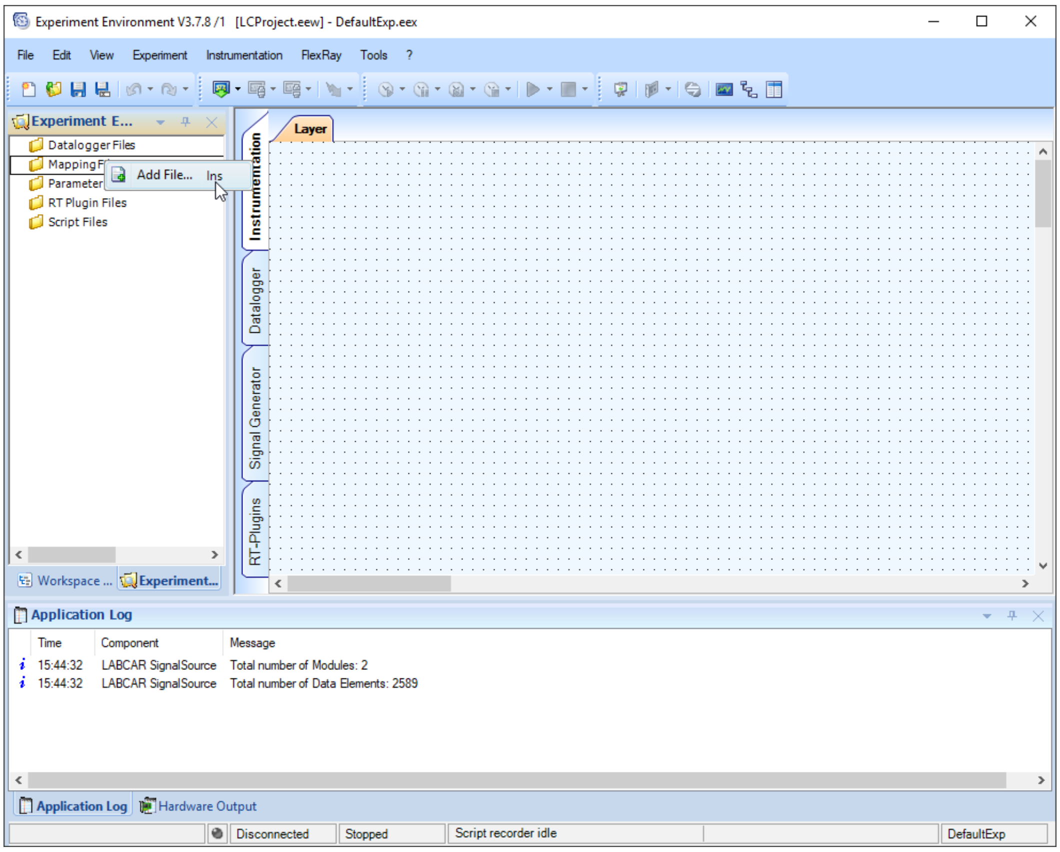


Time	Component	Message
15:44:32	LABCAR SignalSource	Total number of Modules: 2
15:44:32	LABCAR SignalSource	Total number of Data Elements: 2589

Mapping Files

"Mapping files" provide a possibility to simplify the signal handling in LABCAR-EE. For LABCAR-MODEL-VVTB various signal mappings are predefined. This allows a simple access to most important simulation signals without in-depth knowledge of the model.

1. Open **Experiment Explorer.**
2. Add all mapping files located in the install folder at `.\VVTB\Project Items\LCO\VVTB_MT\mapping`.



```
Time      Component            Message
15:44:32  LABCAR SignalSource  Total number of Modules: 2
15:44:32  LABCAR SignalSource  Total number of Data Elements: 2589
```

NOTE

Ensure that the added mapping files are also activated.

Parameter Files

No parameter files are provided. The default values of the Simulink model are used.

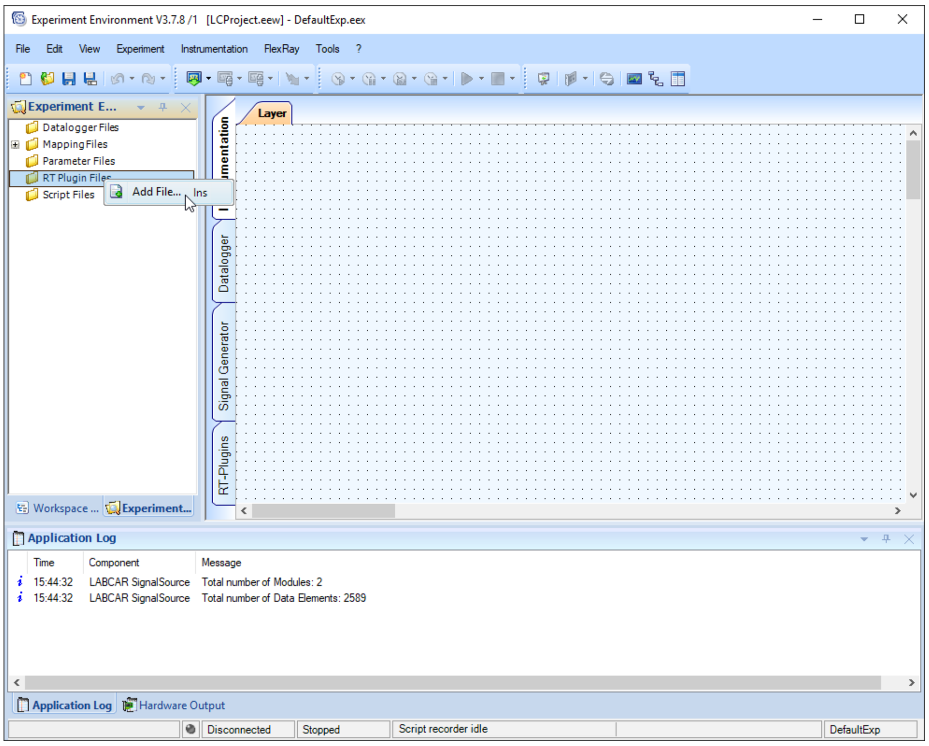
RT-Plugin

RT-Plugins are compiled c-code snippets, which can be added to an experiment without the rebuild of the complete project. It is a tool to add user-defined functionality to an experiment. The GUI of the example project uses a RT-Plugin to extend the functionality of the default instruments of LABCAR-EE.

To add RT-Plugins

1. Open **Experiment Explorer.**
2. Add RT-Plugins located in the install folder at

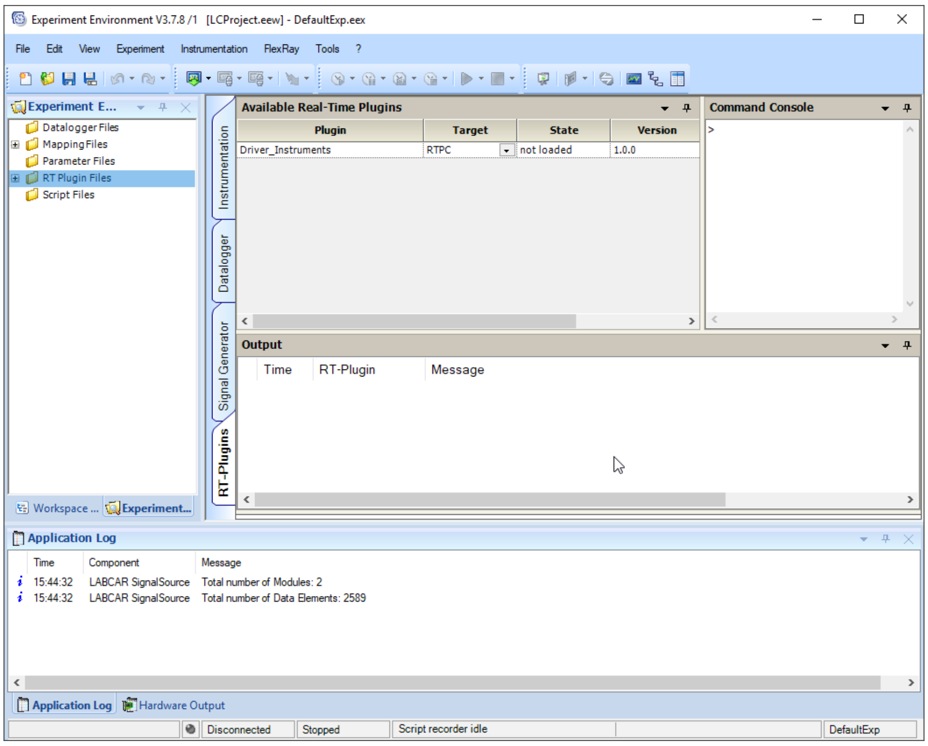
   `.\VVTB\Project Items\LCO\VVTB_MT\rtp\Driver_Instruments.`



Time	Component	Message
15:44:32	LABCAR SignalSource	Total number of Modules: 2
15:44:32	LABCAR SignalSource	Total number of Data Elements: 2589

After the RT-Plugin is added, it is also available in the RT-Plugin tab of LABCAR-EE. Open it to check its availability.

3. Open **RT-Plugins** tab.



Plugin	Target	State	Version
Driver_Instruments	RTPC	not loaded	1.0.0

Time	RT-Plugin	Message

Time	Component	Message
15:44:32	LABCAR SignalSource	Total number of Modules: 2
15:44:32	LABCAR SignalSource	Total number of Data Elements: 2589

NOTE

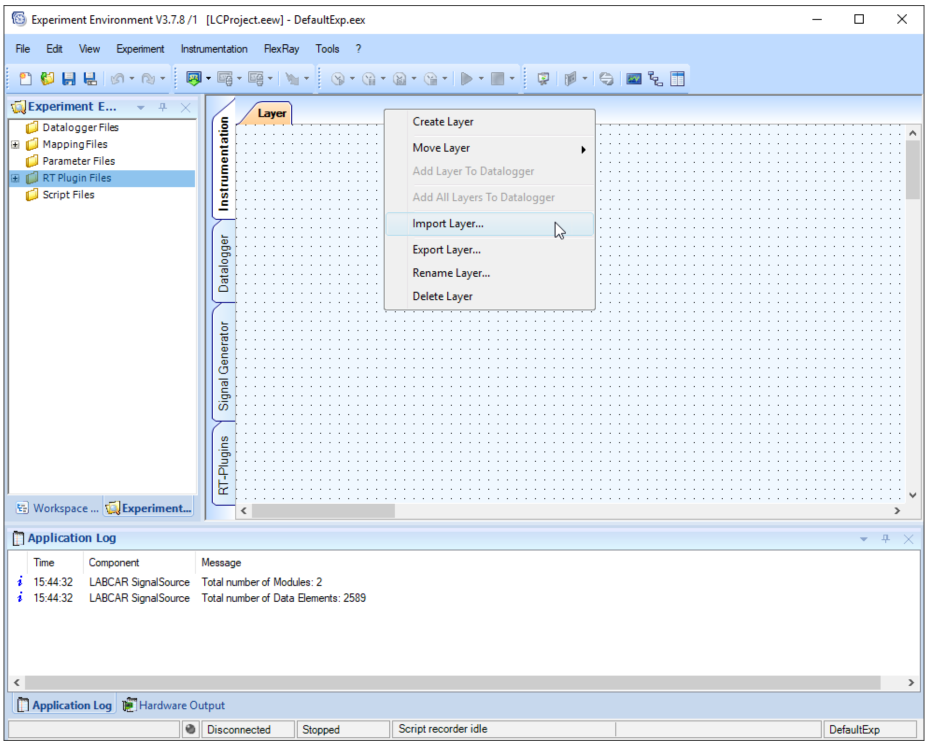
Each RT-Plugin requires a process hook for execution. Ensure that the process hook "Driver_RTPlugin" is available in the OS-settings. See the paragraph “To add RT-Plugins” on page 35.

Layers

The layer provides the basic interface to handle the vehicle model. It allows the setting of the driver mode and the selection of predefined cycles. As preliminary step you can copy images for the layer from the installation folder.

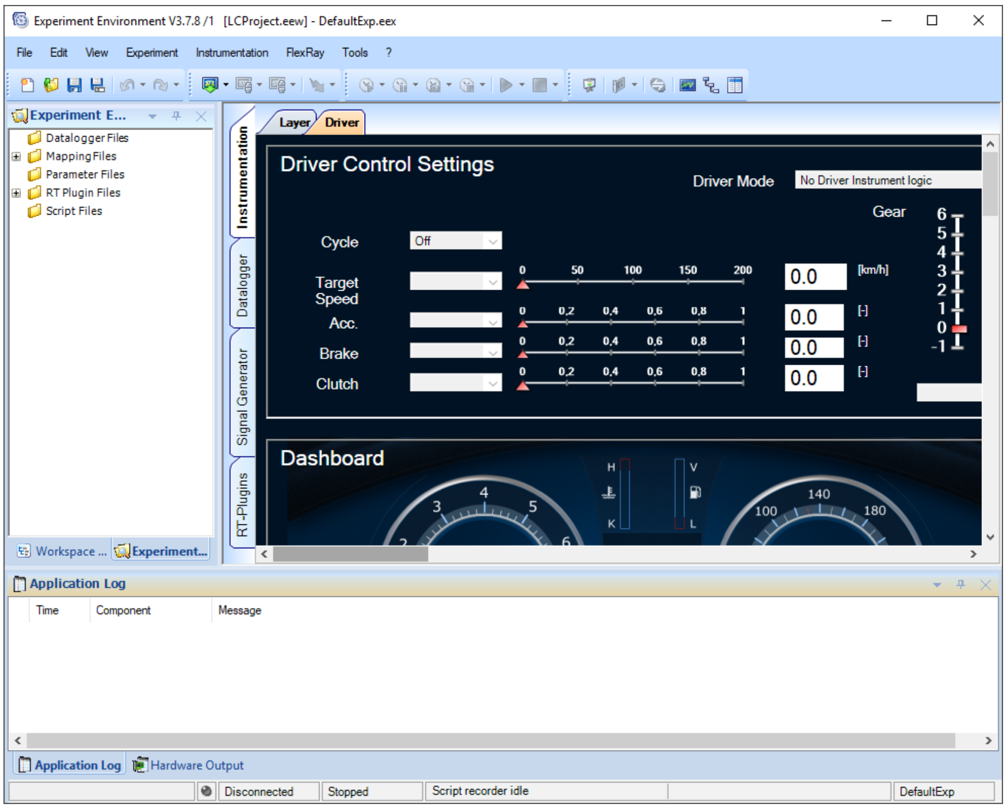
To import layers

1. Copy images from `.\VVTB\Project Items\LCO\VVTB_MT\img`.
2. Create the target folder if it does not already exist in the Experimentation folder and name it as "Image_Files".
3. Paste the images copied in step 1.
4. Import the Driver.layer file as located in `.\VVTB\Project Items\LCO\VVTB_MT\gui.`



```
Time      Component            Message
15:44:32  LABCAR SignalSource  Total number of Modules: 2
15:44:32  LABCAR SignalSource  Total number of Data Elements: 2589
```

> After the successful import of the Driver layer, basic dashboards for handling the VVTB model exist.



NOTE

All predefined instruments use mapped signals provided by the mapping files. For proper execution, ensure that all mapping files are added to the experiment.

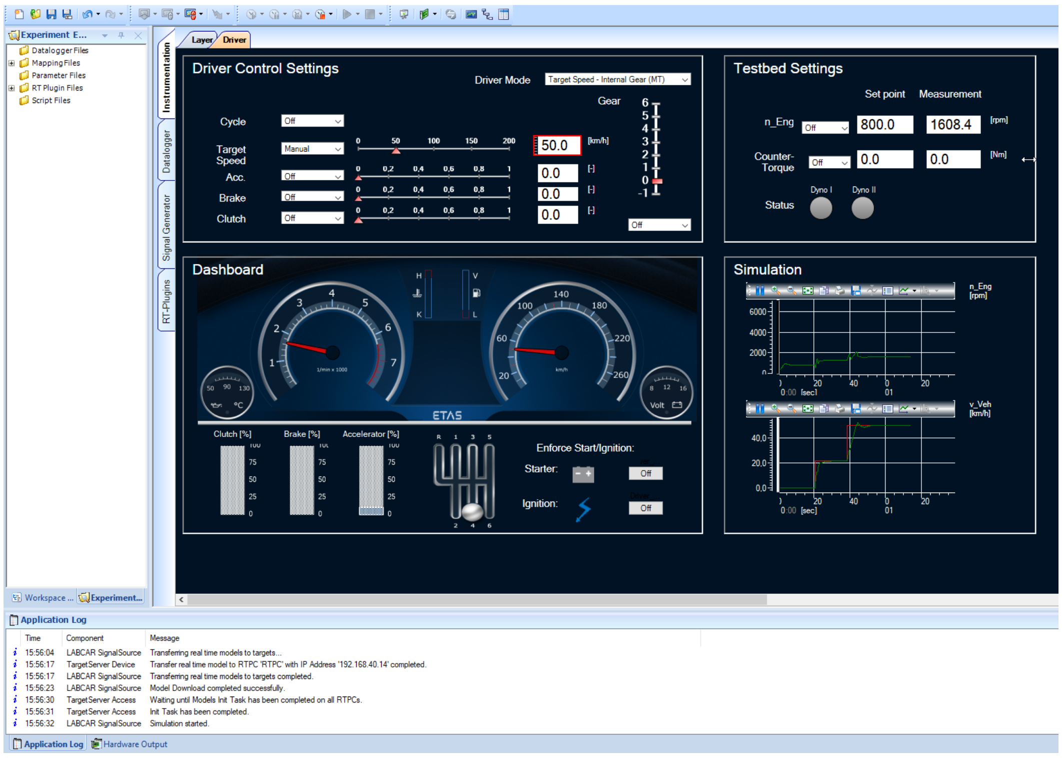
NOTE

To extend the functionality of the standard instruments, so called RT-Plugins are used. Ensure that the pre-compiled RT-Plugins are added to the experiment.

NOTE

Consider that only default LABCAR-EE instruments are used. The layer can be adapted individually to your needs.

All required artifacts for the first execution of the VVTB_MT model are added to the experiment. With that, the experiment is ready for the download to the RTPC and the simulation. The section “Getting Started with the Tutorial Project” on page 42 describes the handling of the simulation model in more detail.



```
Time      Component            Message
15:56:04  LABCAR SignalSource  Transferring real time models to targets...
15:56:17  TargetServer Device  Transfer real time model to RTPC 'RTPC' with IP Address '192.168.40.14' completed.
15:56:17  LABCAR SignalSource  Transferring real time models to targets completed.
15:56:23  LABCAR SignalSource  Model Download completed successfully.
15:56:30  TargetServer Access  Waiting until Models Init Task has been completed on all RTPCs.
15:56:31  TargetServer Access  Init Task has been completed.
15:56:32  LABCAR SignalSource  Simulation started.
```

4.2 Getting Started with the Tutorial Project

The following tutorial gives an introduction on how to use LABCAR-MODEL-VVTB. Within this tutorial LABCAR-MODEL-VVTB is used to represent a vehicle with manual transmission. A physical control unit is not required, since the SoftECU of LABCAR-MODEL-VVTB will be used to control the simple combustion engine model.

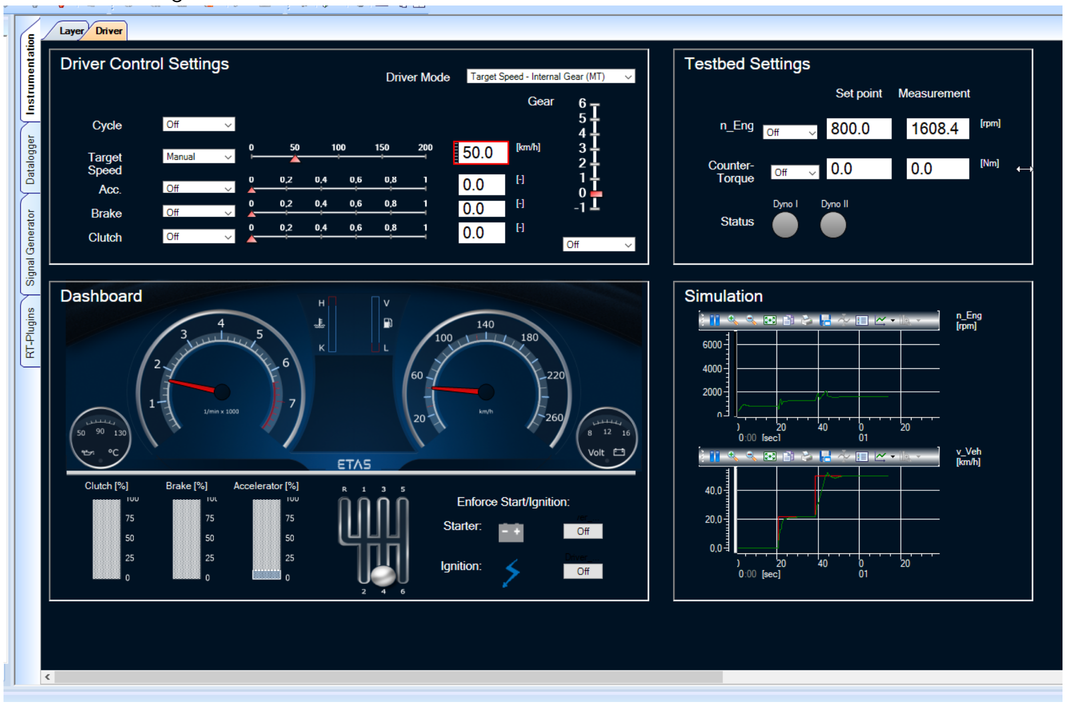
The tutorial project is located in `.\Example Project\LCO\01_VVTB_MT`.

NOTE

Please do not open or manipulate the example project in the installation folder directly. Instead, copy the example project from the installation folder to a location of your choice.

4.2.1 Overview

The "Driver" tab in LABCAR-EE contains four windows as shown in the following figure.



Driver Control Settings

Menu to set the operation mode of the longitudinal driver. Especially, following settings are available:

- Select Driver Mode, e.g. velocity tracking or manual operation
- Select Cycle
- Select inputs for accelerator, clutch, brake, and gear
- Set manual values for accelerator, clutch, brake, and gear

Dashboard

The dashboard shows the current status of the vehicle. It provides typical needle instruments for engine speed and vehicle velocity. Furthermore, it presents the driver output, i.e. the clutch, accelerator, and brake pedal position as well as the gear lever position.

Testbed Settings

LABCAR-MODEL-VVTB provides two engine testbed modes to bring the engine in a particular operation point defined by engine speed and engine torque. The testbed settings allow the parametrization of the desired operation point.

Simulation

Two oscilloscopes provide information about desired and measured engine and vehicle speed.

NOTE

If a different MATLAB version is used than defined in the release note of LAB-CAR-MODEL, the structure of the LABCAR-EE workspace might differ.

4.2.2 Cycle Driving

Cycle driving is one of the most common tests in ECU development. The following guide shows how to drive a pre-defined cycle and a user-defined cycle with LABCAR-MODEL-VVTB.

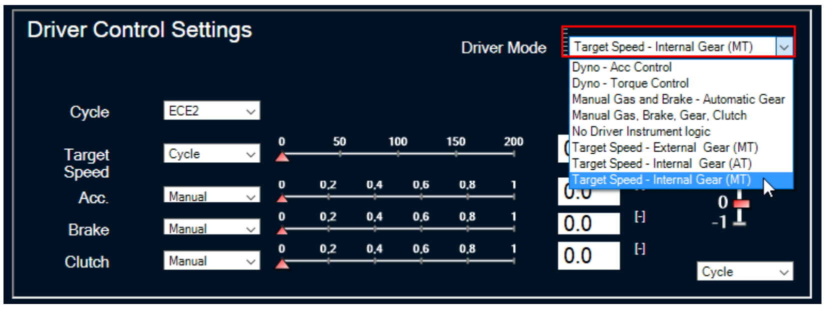
4.2.2.1 Predefined Cycles

LABCAR-MODEL-VVTB provides a set of predefined cycles. They can be accessed by the driver dashboards. In general cycle driving can be defined in two ways:

- For following a pure velocity profile, select:

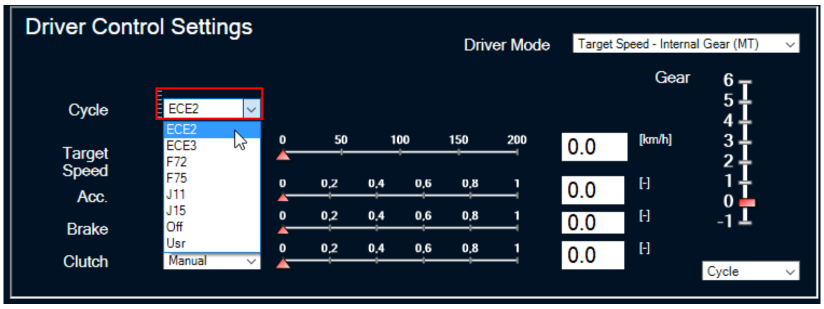
  Set **Driver Mode** → **Target Speed - Internal Gear (MT)**
- For following a velocity and gear profile, select:

  Set **Driver Mode** → T**arget Speed - External Gear (MT**)

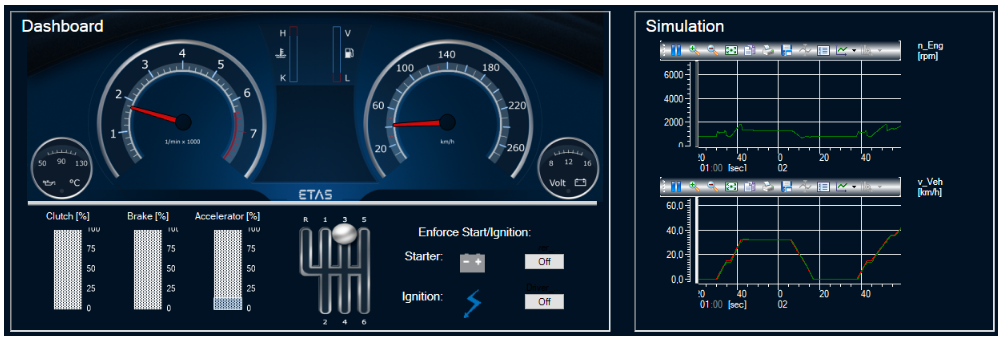

Driver Control Settings
Driver Mode
Target Speed - Internal Gear (MT)
Dyno - Acc Control
Dyno - Torque Control
Manual Gas and Brake - Automatic Gear
Manual Gas, Brake, Gear, Clutch
No Driver Instrument logic
Target Speed - External Gear (MT)
Target Speed - Internal Gear (AT)
Target Speed - Internal Gear (MT)
Cycle
ECE2
Target Speed
Cycle
Acc.
Manual
Brake
Manual
Clutch
Manual
0.0
0.0
0.0
[-]
Cycle

In the given example, the driver determines the gear.

- After that, choose the desired cycle in the drop-down menu:



The Target Speed mode of the driver is automatically set to `Cycle`. Herewith, the driver's reference speed is that one coming from the cycle.



> **NOTE**
>
> When the mode **Target Speed - Internal Gear (MT)** is set, then no manual input for accelerator, brake, or clutch pedal position is possible. The driver model regulates all pedal positions.

4.2.2.2 User-Defined Cycle

User-defined cycles are often used for testing individual functions of an ECU. Such individual cycles are defined via workspace parameters or the signal generator of LABCAR-EE.

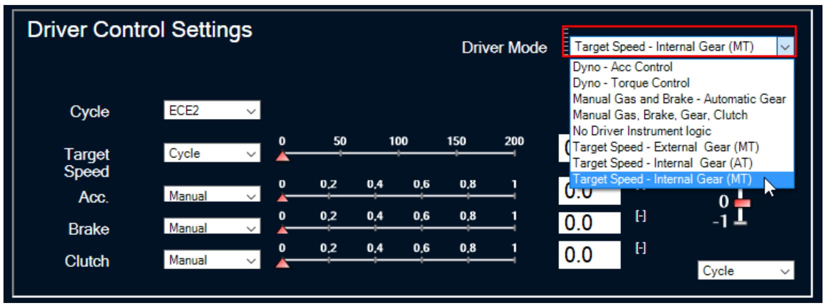
Via Setting Workspace Parameters

The driver model of LABCAR-MODEL-VVTB provides a dedicated location for user-defined cycles. When using that approach, set the desired driver mode.

- For following a pure velocity profile, select:

  Set **Driver Mode** → **Target Speed - Internal Gear (MT)**
- For following a velocity and gear profile, select:

  Set **Driver Mode** → **Target Speed - External Gear (MT)**


Driver Control Settings

Driver Mode: Target Speed - Internal Gear (MT)

Cycle: ECE2

Target Speed: Cycle

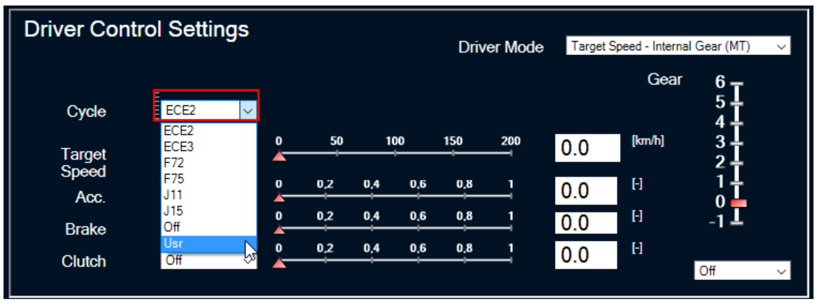
Acc.: Manual

Brake: Manual

Clutch: Manual
Dyno - Acc Control
Dyno - Torque Control
Manual Gas and Brake - Automatic Gear
Manual Gas, Brake, Gear, Clutch
No Driver Instrument logic
Target Speed - External Gear (MT)
Target Speed - Internal Gear (AT)
Target Speed - Internal Gear (MT)
0.0
0.0
[-]
[-]
Cycle

- After that, choose the user cycle in the drop-down menu:

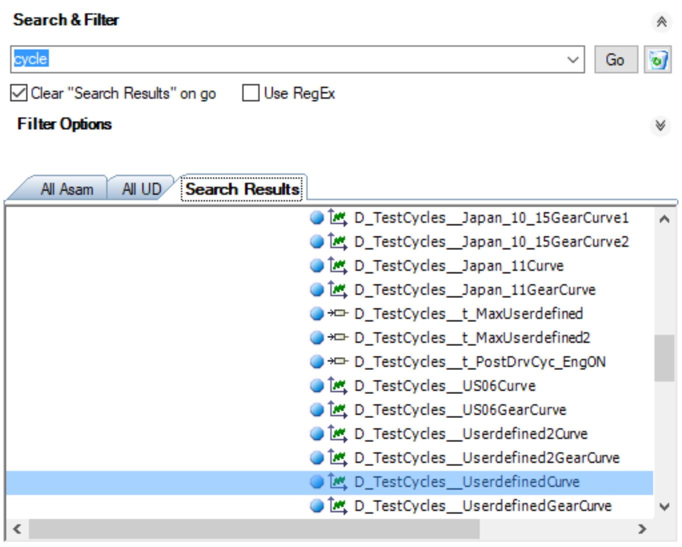
  Set **Cycle Mode** → **Usr**



The user cycle definition is set as 1-D velocity curve over the simulation time. The 1-D curve is accessible via the workspace.

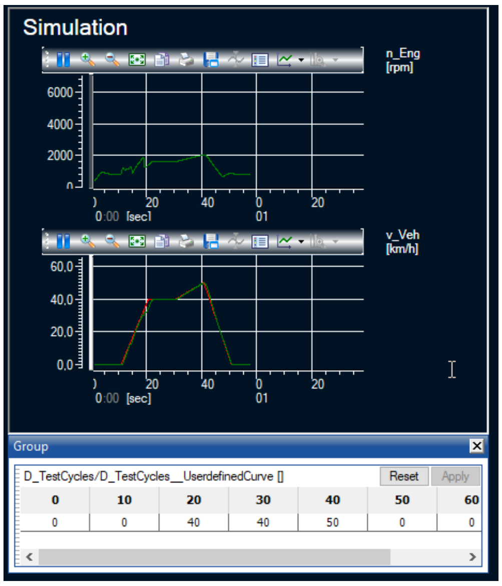
<u>To display the 1-D curve</u>

1. Search `cycle`
2. Drag & Drop `D_TestCycles__UserdefinedCurve`



D_TestCycles__Japan_10_15GearCurve1
D_TestCycles__Japan_10_15GearCurve2
D_TestCycles__Japan_11Curve
D_TestCycles__Japan_11GearCurve
D_TestCycles__t_MaxUserdefined
D_TestCycles__t_MaxUserdefined2
D_TestCycles__t_PostDrvCyc_EngON
D_TestCycles__US06Curve
D_TestCycles__US06GearCurve
D_TestCycles__Userdefined2Curve
D_TestCycles__Userdefined2GearCurve
D_TestCycles__UserdefinedCurve
D_TestCycles__UserdefinedGearCurve

3. Edit `D_TestCycles__UserdefinedCurve`



D_TestCycles/D_TestCycles__UserdefinedCurve []

0	10	20	30	40	50	60
0	0	40	40	50	0	0

The user-defined cycle is defined by the parameter `D_TestCycles__UserdefinedCurve`, which can be adapted accordingly.

4. Start the simulation.

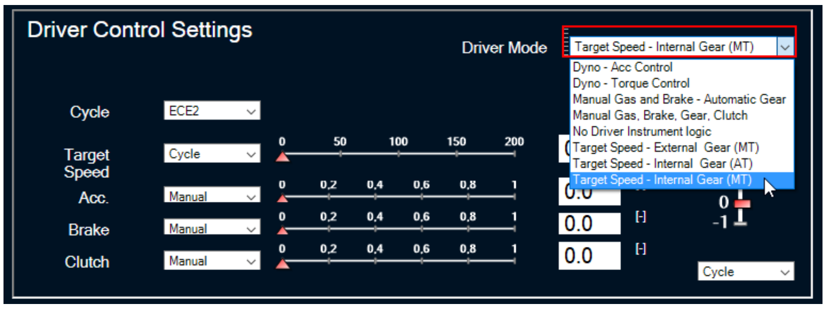
> The user-defined cycle will be followed by the driver.

Via Signal Generator

In the following, a user-defined cycle is created using the "Signal Generator" of LABCAR-EE. Initially, set the desired driver mode.

- For following a pure velocity profile, select:

  Driver Mode → **`Target Speed - Internal Gear (MT)`**



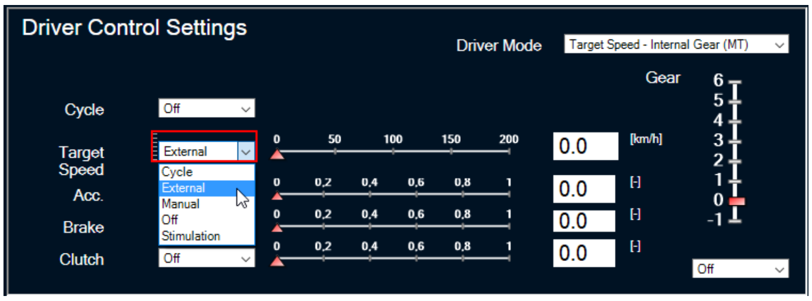
Driver Mode: Target Speed - Internal Gear (MT)

- Dyno - Acc Control
- Dyno - Torque Control
- Manual Gas and Brake - Automatic Gear
- Manual Gas, Brake, Gear, Clutch
- No Driver Instrument logic
- Target Speed - External Gear (MT)
- Target Speed - Internal Gear (AT)
- Target Speed - Internal Gear (MT)

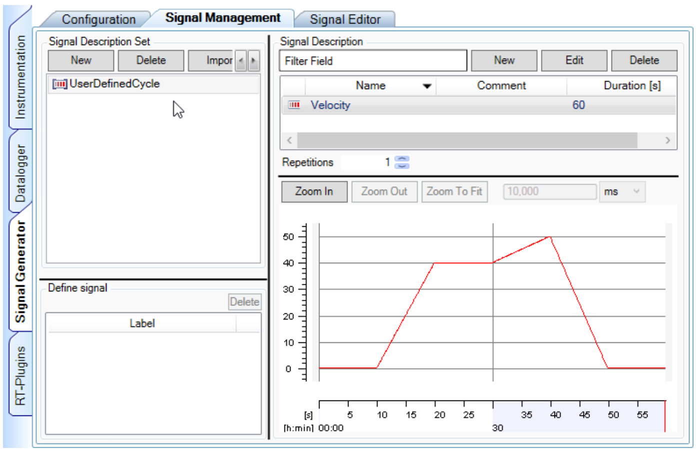
Cycle: ECE2
Target Speed: Cycle
Acc.: Manual 0.0
Brake: Manual 0.0
Clutch: Manual 0.0
Cycle

The driver has individual stimulation inputs for the reference speed. Here, the External stimulation input is selected for connecting the "Signal Generator".

Target Speed → **`External`**

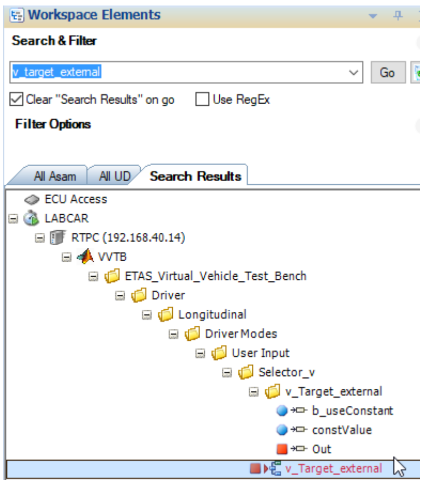


Subsequently, a user-defined cycle with the "Signal Generator" is generated. For further information about the "Signal Generator" see [1].



At that point, a signal generator signal is given. Furthermore, the driver is looking for the reference speed coming from the external stimulation input. The last step is connecting the external stimulation input with the "Signal Generator".

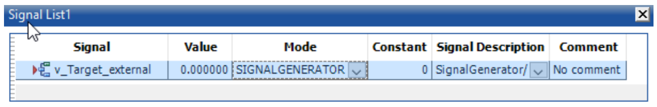
- Search `v_target_external`.
- Right-click → Signal → Trace.



For the traced LCInport, set following options:

- "Mode" → `SIGNALGENERATOR`

- "Signal Description" → `SignalGenerator/...`

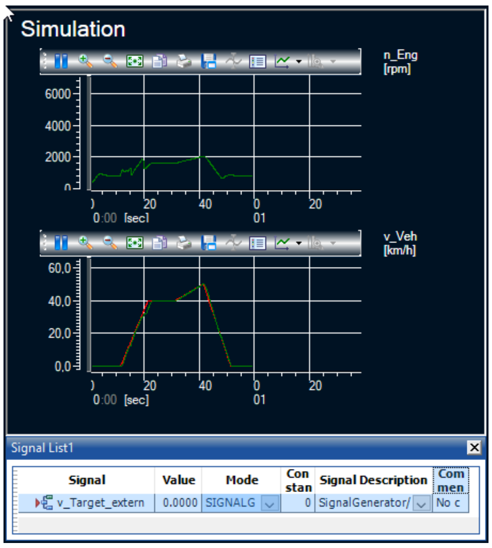


Signal List1

Signal	Value	Mode	Constant	Signal Description	Comment
v_Target_external	0.000000	SIGNALGENERATOR	0	SignalGenerator/	No comment

So the LCPort the driver is looking at is connected to the "Signal Generator".

- Start the simulation and the driver will follow the user-defined cycle.



Signal List1

Signal	Value	Mode	Con stan	Signal Description	Com men
v_Target_extern	0.0000	SIGNALG	0	SignalGenerator/	No c

4.3 Signal Interface

All of the presented interfaces connect driver, SoftECU, and vehicle in the example project of the section “Project and Experiment Creation” on page 28. They are the minimal interfaces that a SoftECU or a physical ECU has to satisfy when running LABCAR-MODEL-VVTB.

The model reference guide of LABCAR-MODEL-VVTB provides the complete documentation of the physical interfaces of each S-Fcn and sub-system.

4.3.1 Driver

Inputs

Name	Type	Unit	Description
-	-	-	-

Outputs

Name	Type	Unit	Description
b_Ignition	Bool	-	Driver ignition request
b_Starter	Bool	-	Driver starter request
i_Gear	Int	-	Desired gear
r_Accelerator_0_1	Double	-	Accelerator pedal position
r_Brake_0_1	Double	-	Brake pedal position
r_Clutch_0_1	Double	-	Clutch pedal position

4.3.2 SoftECU

Inputs

Name	Type	Unit	Description
Starter	Bool	-	Received starter signal
n_Engine	Double	rpm	Measured engine speed
M_Ind_In	Double	Nm	Measured induced engine torque
M_Drag_In	Double	Nm	Measured engine drag torque
M_CombEngine	Double	Nm	Effective engine torque
N_ThrottlePlateOpening_r_0_100	Double	%	Measured throttle plate position
In_Gearbox_Shifting	Bool	-	Flag indicating active gear box shifting
Ignition	Bool	-	Driver ignition flag
Gear	Int	-	Current gear
Clutch	Double	-	Clutch pedal position
Brake	Double	-	Brake pedal position
Battery	Bool	-	Battery on / off
Accelerator	Double	-	Accelerator pedal position

Outputs

Name	Type	Unit	Description
Accelerator_Out	Double	-	Processed accelerator pedal position
Brake_Out	Double	-	Processed brake pedal position
Clutch_Out	Double	-	Processed clutch pedal position
Gear_Out	Int	-	Processed gear
Ignition_Out	Bool	-	Processed ignition flag
M_Target_fast_Out	Double	Nm	Desired torque for fast requests
M_Target_slow_Out	Double	Nm	Desired torque for slow re-quests
Starter_Out	Bool	-	Processed starter flag
SignalBit_slow_out	Bool	-	Flag indicating fast torque request
SignalBit_fast_out	Bool	-	Flag indicating slow torque request

4.3.3 Vehicle/Drivetrain

Inputs

Name	Type	Unit	Description
CU_Gear	Int	-	Requested gear

Outputs

Name	Type	Unit	Description
Actual_Gear	Int	-	Current gear
GearboxShifting	Bool	-	Flag indicating active gear box shifting
n_Engine	Double	rpm	Calculated engine speed

4.3.4 Vehicle/CombustionEngine

Inputs

Name	Type	Unit	Description
SignalBit_slow	Bool	-	Flag indicating slow torque request
SignalBit_fast	Bool	-	Flag indicating fast torque request
M_Target_slow	Double	-	-
M_Target_fast	Double	-	-

Outputs

Name	Type	Unit	Description
M_Drag	Double	Nm	Calculated engine drag torque
M_Engine	Double	Nm	Calculated engine torque
M_Ind	Double	Nm	Calculated induced engine torque
M_Ind_Max	Double	Nm	Calculated maximum induced engine torque
ThrottlePlateOpening_r_0_100	Double	-	Throttle plate opening

4.3.5 Driver/BodyComponents

Inputs

Name	Type	Unit	Description
CU_Starter	Bool	-	Starter status communicated by CU
CU_Ignition	Bool	-	Ignition status communicated by CU

Outputs

Name	Type	Unit	Description
BatteryIsOn	Bool	-	Battery status

NOTE

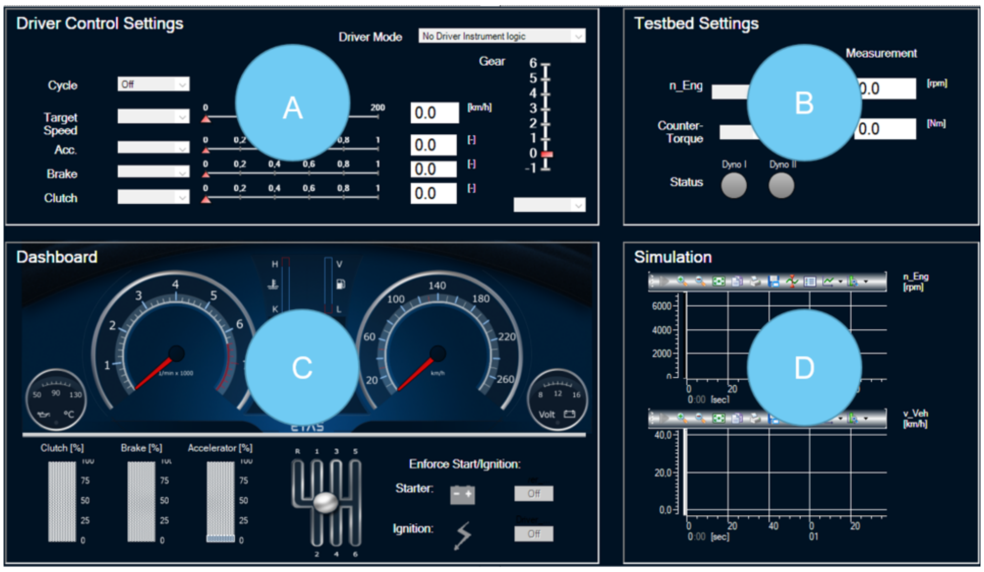
Interfaces of the physical subsystems, e.g., drivetrain, combustion engine or chassis, are documented in the model reference guide of LABCAR-MODEL-VVTB.

4.4 User Interface

The provided Graphical User Interface (GUI) for LABCAR-MODEL-VVTB contains basic functionality to operate the vehicle model comfortably. The GUI allows to select a driver mode and to define external trajectories to be followed by the vehicle model. The driver mode defines the operational mode of the vehicle model via its included model of a driver. Basic observables about the vehicle's operation and state are presented graphically.

4.4.1 Overview

The GUI consists of the 4 sections in the following figure:



A The section "Driver Control Settings" provides controls to operate the vehicle. The available functionality depends on the chosen driver mode. Driver modes are described in the section “Driver Modes” on page 54. After selecting the driver mode, you can control the vehicle by interacting with the available controls. The individual control elements are described in the section “Controlling the Vehicle” on page 56.

B "Testbed Settings" are an extension of the Driver Control Settings to simulate the vehicle sitting on a roller dynamometer test bench. Testbed Settings are enabled in specific *Dyno* driver modes only.

C The "Dashboard" presents the current engine speed and the vehicle speed similar to a real car's cockpit. Additionally, the positions of the clutch pedal, the brake pedal, the accelerator pedal and the gear lever are shown. 0% indicates an untouched pedal, 100% indicates a fully pressed pedal. Icon indicators of the starter's and the ignition's state complement the Dashboard. There are On/Off switches that enforce the activation of starter and ignition if set to On. If set to off, starter and ignition are handled by the model.

D The section "Simulation" contains graphs of the engine speed and the vehicle speed over time.

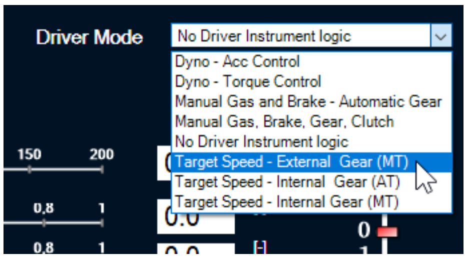
NOTE

The GUI uses standard instruments of LABCAR-EE and an open-source RT-Plugin to extend the instruments' functionality. This can be used as template for individual GUIs easily.

For the GUI to function as expected, its RT-Plugin `Driver_Instruments.rtp` and the mappings `Driver_Instruments.txt` need to be activated as described in the section "Project and Experiment Creation" on page 28.

4.4.2 Driver Modes

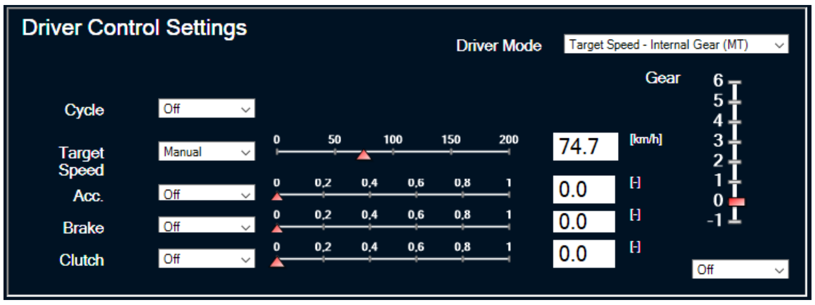
The GUI's Driver Control Settings section contains a drop-down list labeled Driver Mode. Using this drop-down list, you can select different driver modes or switch between them:


Driver Mode
No Driver Instrument logic
Dyno - Acc Control
Dyno - Torque Control
Manual Gas and Brake - Automatic Gear
Manual Gas, Brake, Gear, Clutch
No Driver Instrument logic
Target Speed - External Gear (MT)
Target Speed - Internal Gear (AT)
Target Speed - Internal Gear (MT)
150
200
0,8
1
0,8
1
0.0
0

Switching between driver modes changes the behavior of the vehicle's driver model. Parts of the GUI are activated or deactivated depending on the selected driver mode. This activation or deactivation of controls can take a fraction of a second depending on the Windows host machine and LABCAR.

The following list describes the available driver modes, the resulting behavior of the driver model and the available GUI control elements in every mode. These individual control elements are explained in the section "Controlling the Vehicle" on page 56.

Target Speed - Internal Gear (MT)


Driver Control Settings

Driver Mode: Target Speed - Internal Gear (MT)

Cycle: Off

Target Speed: Manual — 74.7 [km/h]

Acc.: Off — 0.0 [-]

Brake: Off — 0.0 [-]

Clutch: Off — 0.0 [-]

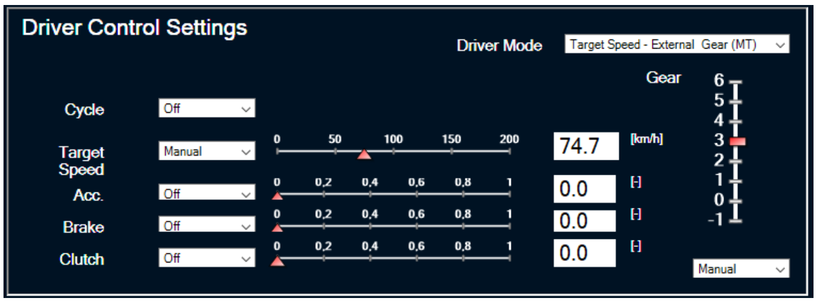
Gear: Off

This mode is intended for the use in the model variant VVTB-MT, the variant with manual transmission. When this mode is selected, all GUI control elements except Target Speed are disabled. Using the slider or the input field, a target speed can be defined. The driver will accelerate or decelerate the vehicle to the defined target speed. It will handle the accelerator, brake and clutch pedals as well as shift gears automatically.

Target Speed - Internal Gear (AT)

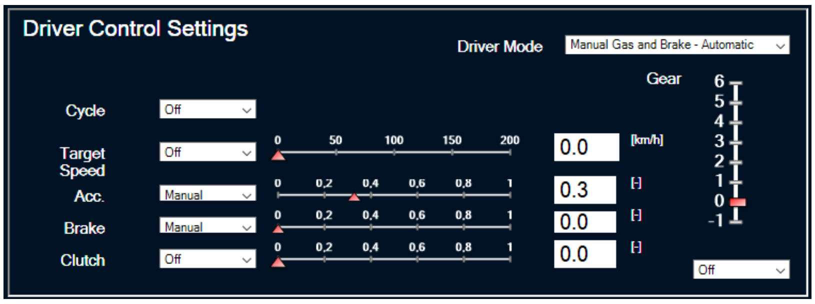
This mode is intended for the use in the model variant VVTB-AT, the variant with automatic transmission. The GUI functionality is identical to `Target Speed - Internal Gear (MT)`.

Target Speed - External Gear (MT)


Driver Control Settings
Driver Mode: Target Speed - External Gear (MT)
Cycle: Off
Target Speed: Manual 74.7 [km/h]
Acc.: Off 0.0 [-]
Brake: Off 0.0 [-]
Clutch: Off 0.0 [-]
Gear: 3, Manual

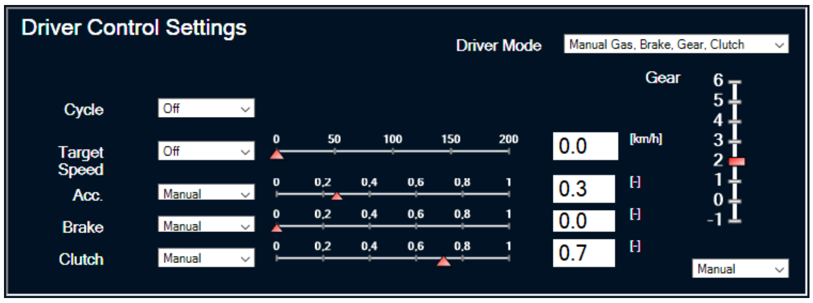
This mode is similar to the `Target Speed – Internal Gear` modes. The model will still operate the pedals automatically to reach the user-defined target speed. However, the model-internal determination of the appropriate gear is disabled. The driver has control over the gear lever at the right of the GUI section shown above. All other controls except target speed and gear are disabled since they are handled by the driver model.

Manual Gas and Brake - Automatic Gear


Driver Control Settings
Driver Mode: Manual Gas and Brake - Automatic
Cycle: Off
Target Speed: Off 0.0 [km/h]
Acc.: Manual 0.3 [-]
Brake: Manual 0.0 [-]
Clutch: Off 0.0 [-]
Gear: 0, Off

In this mode, the driver model does not operate the accelerator and the brake. Instead, the user has control over these two pedals. However, the driver model still shifts gears and operates the clutch automatically. This driver mode operates the vehicle as if it was an automatic car.

Manual Gas, Brake, Gear, Clutch

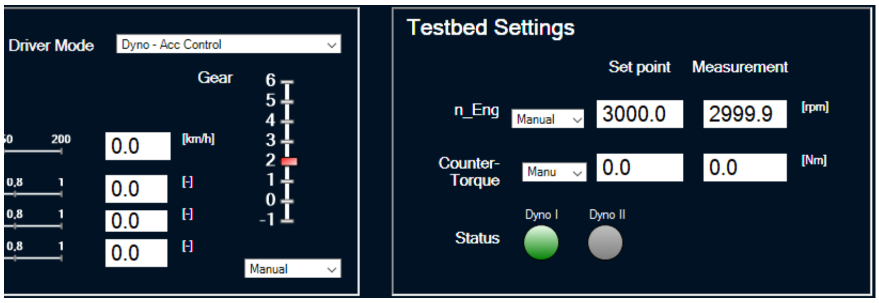

Driver Control Settings
Driver Mode: Manual Gas, Brake, Gear, Clutch
Cycle: Off
Target Speed: Off 0.0 [km/h]
Acc.: Manual 0.3 [-]
Brake: Manual 0.0 [-]
Clutch: Manual 0.7 [-]
Gear: 2, Manual

This mode offers full manual control over all three pedals and the gear lever to the user and deactivates the driver model.

NOTE

The engine is easily stalled in this mode because of the user-defined clutch control.

Dyno - Acc Control


Driver Mode: Dyno - Acc Control

Gear: 6 5 4 3 2 1 0 -1 — Manual

200	0.0	[km/h]
0,8 1	0.0	[-]
0,8 1	0.0	[-]
0,8 1	0.0	[-]

Testbed Settings

		Set point	Measurement	
n_Eng	Manual	3000.0	2999.9	[rpm]
Counter-Torque	Manu	0.0	0.0	[Nm]

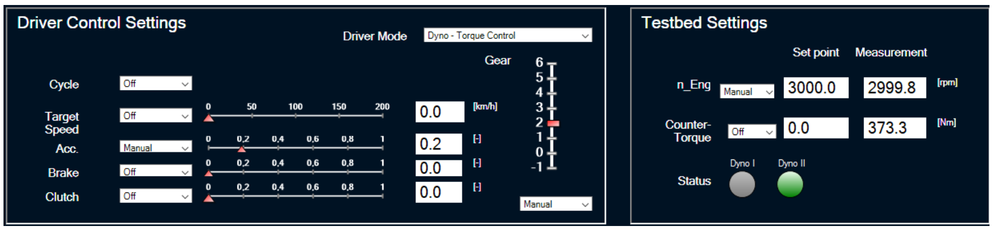
Status: Dyno I ●, Dyno II ●

This mode enables the GUI section "Testbed Settings". When switching to this Dyno mode, the desired gear has to be chosen. Under "Set point", you can define a desired engine speed in the "n_Eng " row. The driver model will then accelerate the vehicle as necessary to reach the set point engine speed. This may take a few seconds. The green light "Dyno I" indicates that the engine speed approaches the set point. An additional counter-torque can be applied manually as well in the same "Set point" column in the "Counter-torque" row.

NOTE

Before activating this mode, the vehicle needs to have an appropriate speed first, because it relies on a manual gear choice. This can be achieved. e.g., by using one of the "Target Speed" driver modes.

Dyno - Torque Control


Driver Control Settings

Driver Mode: Dyno - Torque Control

Cycle	Off			
Target Speed	Off	0 50 100 150 200	0.0	[km/h]
Acc.	Manual	0 0,2 0,4 0,6 0,8 1	0.2	[-]
Brake	Off	0 0,2 0,4 0,6 0,8 1	0.0	[-]
Clutch	Off	0 0,2 0,4 0,6 0,8 1	0.0	[-]

Gear: 6 5 4 3 2 1 0 -1 — Manual

Testbed Settings

		Set point	Measurement	
n_Eng	Manual	3000.0	2999.8	[rpm]
Counter-Torque	Off	0.0	373.3	[Nm]

Status: Dyno I ●, Dyno II ●

This mode enables the GUI section "Testbed Settings". When switching to this Dyno mode, the desired gear has to be chosen by the user, as well as a fixed accelerator pedal position. Under "Set point", a desired engine speed can be defined. The model will then apply counter torque on the rollers as necessary, until the set point engine speed is reached. This may take a few seconds. The green light "Dyno II" indicates that the engine speed approaches the set point.

NOTE

Before activating this mode, the vehicle needs to have an appropriate speed first, because it relies on a manual gear choice. This can be achieved, e.g., by using one of the "Target Speed" driver modes.

4.4.3 Controlling the Vehicle

The GUI provides individual control elements to reflect the controls that are available in a real vehicle:

- accelerator pedal

- brake pedal
- clutch pedal
- gear lever

Since LABCAR-MODEL-VVTB contains a complex driver model, which is able to operate these elements automatically, interaction possibilities depend on the selected driver mode. For the "Target Speed" driver modes, additional controls are available instead of the pedals:

- Target Speed
- Cycle

The usage of these two controls is special and explained in the section “Driving Test Cycles” on page 59.

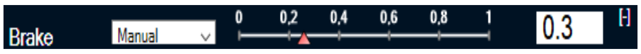
A typical control element consists of:

- Input selector:

  a drop-down list to select the input source of the control's value.
- Slider:

  a graphical interaction method to change the control's value manually.
- Input field:

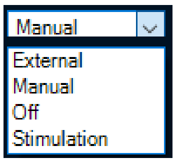
  a decimal input field to change the control's value manually.



Input Selector

The drop-down list of every control element allows to select different input sources for the control's value. Every interface possibility that LABCAR provides can be connected as input source and switched between, using the input selector. The user can switch the input source at any time during the simulation.

The available input sources are:



- `Off`

  A control that is unavailable to the user in the selected driver mode is set to `Off`. This option is not selectable by the user. You have to change the driver mode to use the respective control. A control set to `Off` is either controlled automatically by the driver model or unavailable in the selected driver mode. It is set automatically by selecting the driver mode.
- `Manual`

  A control that is available to the user in the selected driver mode is by default set to `Manual`. This option allows you to change the control's value using the slider or the input field.

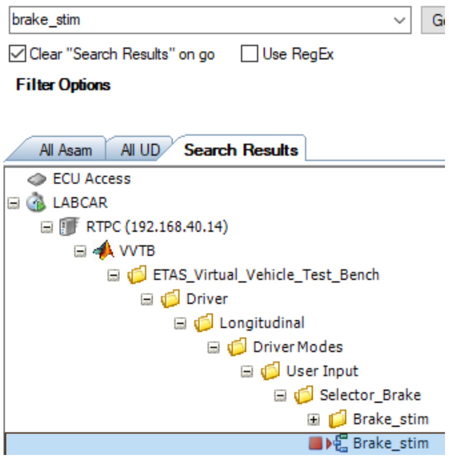
- `Stimulation`

  Each control element has a dedicated LCInport inside the model that can be used as input source for the control's value. When the option `Stimulation` is selected instead of `Manual`, the respective LCInport in the model is used as input source instead of the GUI slider/input field.

  This LCInport is intended for connection to a "SignalGenerator" or connection to another module/model of the LABCAR project.

  The LCInports of every control can be found in the LABCAR-EE Workspace Elements tree under:

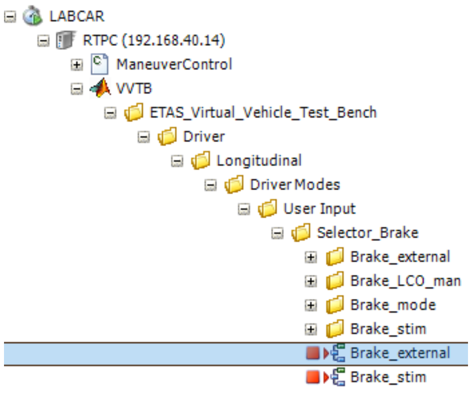
  `VVTB/ETAS_Virtual_Vehicle_Test_Bench/Driver/Longitudinal/Driver Modes/User Input/Selector_<ControlName>/<ControlName>_stim:`

  For more information on LCInports and SignalGenerators refer to [1].

- `External`

  Like `Stimulation`, the option `External` uses an LCInport as input source. It is available next to the Stimulation LCInport:

  When the `Stimulation` LCInport is already connected to a SignalGenerator, this `External` LCInport could be used to connect to a hardware controller or another controller as well. The drop-down option `External` can then be used to switch the input source from `Manual` or `Stimulation` to `External` "on-the-fly".

Slider

The slider can be used to change the control's value using the mouse cursor. For the accelerator, brake and clutch pedals, 0 represents an untouched pedal and 1 represents a fully pressed pedal.

The slider's value is only respected by the model when the input selector is set to `Manual`.

Input Field

The input field can be used to define a precise decimal value. The display is rounded by default, but the number of displayed decimals can be changed using the LABCAR-EE "Properties" window.

The input field's value is only respected by the model when the input selector is set to `Manual`.

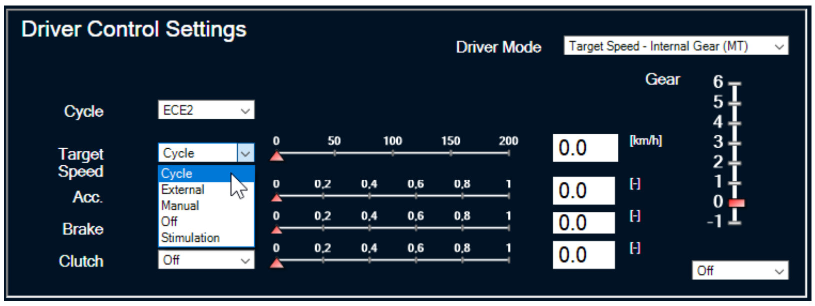
4.4.3.1 Driving Test Cycles

The driver model of LABCAR-MODEL-VVTB is able to follow predefined test cycles or a user-defined test cycle without any user interaction. A test cycle is defined as a target vehicle speed trajectory over time. The following test cycles are available:

- ECE2
- ECE3
- F72
- F75
- J11
- J15
- Usr

Usr does not contain a test cycle by default. This allows you to define a test cycle that can then be executed in the same fashion as the predefined test cycles.

Test cycles can be executed when one of the Target Speed driver modes is selected. The Target Speed's input selector is similar to the other controls' input selectors, but offers the additional option Cycle:



Driver Control Settings

Driver Mode: Target Speed - Internal Gear (MT)

Cycle: ECE2

Target Speed: Cycle — 0.0 [km/h]

Acc.: 0.0 [-]

Brake: 0.0 [-]

Clutch: Off — 0.0 [-]

Gear: Off

When "Target Speed" is set to `Cycle`, the drop-down list labeled "Cycle" can be used to select one of the test cycles listed above. The driver will then drive that test cycle.

NOTE

The functionality uses a model-internal clock, which starts running at simulation start. To execute a test cycle from its start, it is recommended to select the test cycle directly after simulation start.

When "Target Speed" is set to any other input source than `Cycle`, the drop-down list "Cycle" is disabled and labeled `Off`. A previously selected test cycle will be remembered when switching "Target Speed" back to `Cycle`.

The section “Getting Started with the Tutorial Project” on page 42 contains a tutorial showing the process of defining and executing test cycles.

5 LABCAR-MODEL-ICE

ETAS LABCAR-MODEL-ICE is a detailed model of an internal combustion engine (ICE) and of its main subsystems. These are the combustion system, the fuel system, the intake and exhaust system, and the aftertreatment system. The model includes several preconfigured standard variants for Diesel, CNG, and Gasoline engines.

5.1 Project and Experiment Creation

In the following, step-by-step guides describe the generation of a COSYM project or a LABCAR-OPERATOR project using LABCAR-MODEL-ICE. For the execution of the project, no physical ECU is needed. Instead, a SoftECU controls the internal combustion engine model. The project to be created contains the manual transmission vehicle model LABCAR-MODEL-VVTB given by the VVT-B_MT_ICE model configuration that is combined with the gasoline variant of LABCAR-MODEL-ICE given by the ICE_G_1B_1Stg_DEMO model configuration.

5.1.1 Project Creation in COSYM

This section addresses qualified personnel with knowledge in using COSYM. If you do not have any experience in using COSYM, please read first [6].

For the manual project creation using LABCAR-MODEL-ICE with COSYM follow the steps described in this section.

<u>To set up the COSYM project</u>

1. Open the COSYM application.

   The COSYM application can already have a open project. In this case

2. Click **File** → **New** → **Project.**

or

   If there is no previous project loaded

   - Click **Create new project**.

3. Choose the location for the project.and name the project.
4. Click **Create.**

NOTE

Refer to section 7.1 in [6] for additional help regarding the project creation in COSYM.

> The "Library" section of the project is empty to start with. A default system is created with the name "System" and automatically mapped to "HiL_Target".

<u>To add a hardware module to the system</u>

1. Double click on the pane "System".
2. Expand "System"
3. Right click on "HiL_Target" and select **Create RTIO module** from the drop-down menu.

> The "System" shows the added hardware module in "Diagram" view.

The full vehicle model as well as the internal combustion engine model consist each out of a single Simulink module. This module also contains driver, environment, vehicle, and SoftECU, which mimics the behavior of an EMS. It will control the internal combustion engine model.

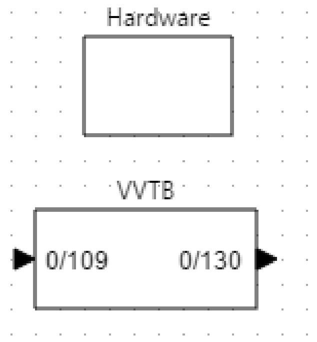
To add the virtual vehicle test bench model

1. In the "Library" pane, right click on the "Control" folder and choose "Import Model".
2. Under "Source of the model to be imported", click on **Browse** button.
3. Select the full vehicle model configuration VVTB_MT_ICE. The according LMD file "`VVTB.lmd`", which internally refers to the model configuration `VVTB.mdl`. The corresponding LMD file "`VVTB.lmd`" is located in the installation folder:

   ```
   .\VVTB\Project Items
   \COSYM\VVTB_MT_ICE\mdl\Simulink_VVTB\src\VVT
   B_LABCAR_rtw.
   ```
4. Click on the **Import** button.

   The model "VVTB" is added to the "Library" under the "Control" folder.
5. Drag and drop the "VVTB" model from the "Library" pane
   - to the "Diagram" area of the '"System"
   - and to the "Units" folder of "System" under the "Systems" pane.

> The "System" shows the added module in the "Diagram" view.



NOTE

The Simulink model references additional dependencies by using the "`ExternalFiles.xml`". Please see section “Mapping External Files for HiL Execution” on page 18 for further details.

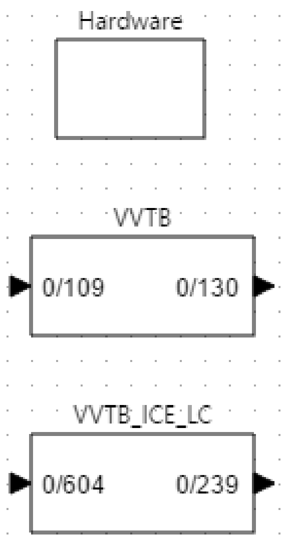
To add the internal combustion engine model

1. In the "Library" pane, right click on the "Control" folder and choose "Import Model".

2. Under "Source of the model to be imported", click on **Browse** button.
3. Select the internal combustion engine model configuration ICE_G_1B_1Stg_DEMO. The according LMD file `VTB_ICE_LC.lmd`, which internally refers to model configuration `VVTB_ICE_LC.mdl`. The corresponding LMD file `VVTB.lmd` is located in the installation folder: `.\ICE\Project Items \COSYM\ICE_G_1B_1Stg_DEMO\mdl\Simulink_VVTB_ICE_LC\src\VVTB_ICE_LC_LABCAR_rtw.`
4. Click on the **Import** button.

   The model "VVTB_ICE_LC" is added to the "Library" under the "Control" folder.
5. Drag and drop the "VVTB" model from the "Library" pane
   - to the "Diagram" area of the '"System"
   - and to the "Units" folder of "System" under the "Systems" pane.

> The "System" shows the added module in the "Diagram" view.



NOTE

The Simulink model references additional dependencies by using the "`ExternalFiles.xml`". Please see section “Mapping External Files for HiL Execution” on page 18 for further details.

The internal combustion engine model requires initial environment conditions for proper simulation. For this purpose, C-Code modules are used for temperature and pressure initialization.

The following C-Code modules have to be added:

- `Init_T_Ambient.lmd`

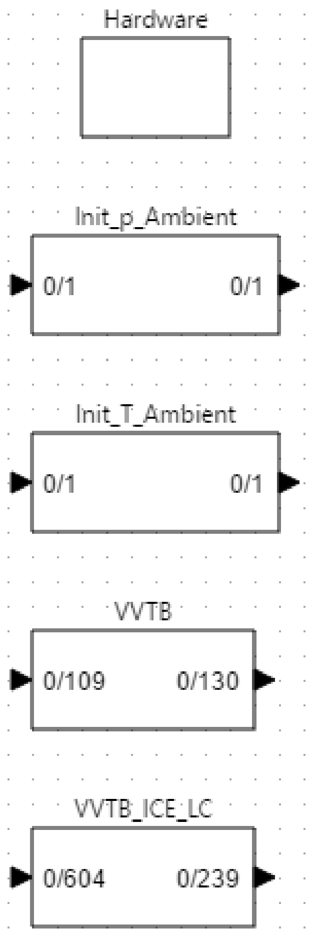
  Located in installation folder at
  `.\ICE\Project Items`
  `\COSYM\ICE_G_1B_1Stg_DEMO\cmod\Init_T_Ambient`
- `Init_p_Ambient.lmd`

  Located in installation folder at
  `.\ICE\Project Items`
  `\COSYM\ICE_G_1B_1Stg_DEMO\cmod\Init_p_Ambient`

<u>To add a C-Code module</u>

1. In the Library pane, right click on the "Environment" folder and choose "Import Model".
2. Under "Source of the model to be imported", click on the **Browse** button.
3. Select the C-Code module's LMD file from the locations specified above.
4. Click on the **Import** button.
5. The C-Code model is added to the "Library" under the "'Environment" folder.
6. Open the "System" from the "Systems" pane by double click.
7. Drag and drop the C-Code model from the "Library" pane
   - to the Diagram area of the "System".
   - to the "Units" folder of "System" under the "Systems" pane.
8. Repeat the steps 1 to 7 for the 2 C-Code modules listed above.

> The "System" shows the added C-Code module in the "Diagram" view.



By default, no model connections are established. For example the SoftECU is not connected to the vehicle, driver model or engine model, although it is part of the already integrated Simulink module `VVTB.mdl`. Pre-defined connection files couple the vehicle model, the engine model and the SoftECU.

The following connection files can be added:

- `BodyComponents_Basic-CU_EMS.xml`
- `CU_EMS-Vehicle.xml`
- `CU_Testbed-VehicleDynamics_Longitudinal.xml`
- `Driver_Longitudinal-CU_Testbed.xml`
- `Driver-CU_EMS.xml`
- `Driver-CU_TCU_MT.xml`
- `Drivetrain_Longitudinal-CU_EMS.xml`
- `Drivetrain_Longitudinal-CU_TCU_MT.xml`
- `Drivetrain_Longitudinal-CU_Testbed.xml`

  These above files are located in installation folder at `.\VVTB\Project Items\COSYM\VVTB_MT_ICE\conn`
- `CombustionEngine_G-VVTB_ICE.xml`
- `CombustionEngine-ICE_CU_EMS.xml`

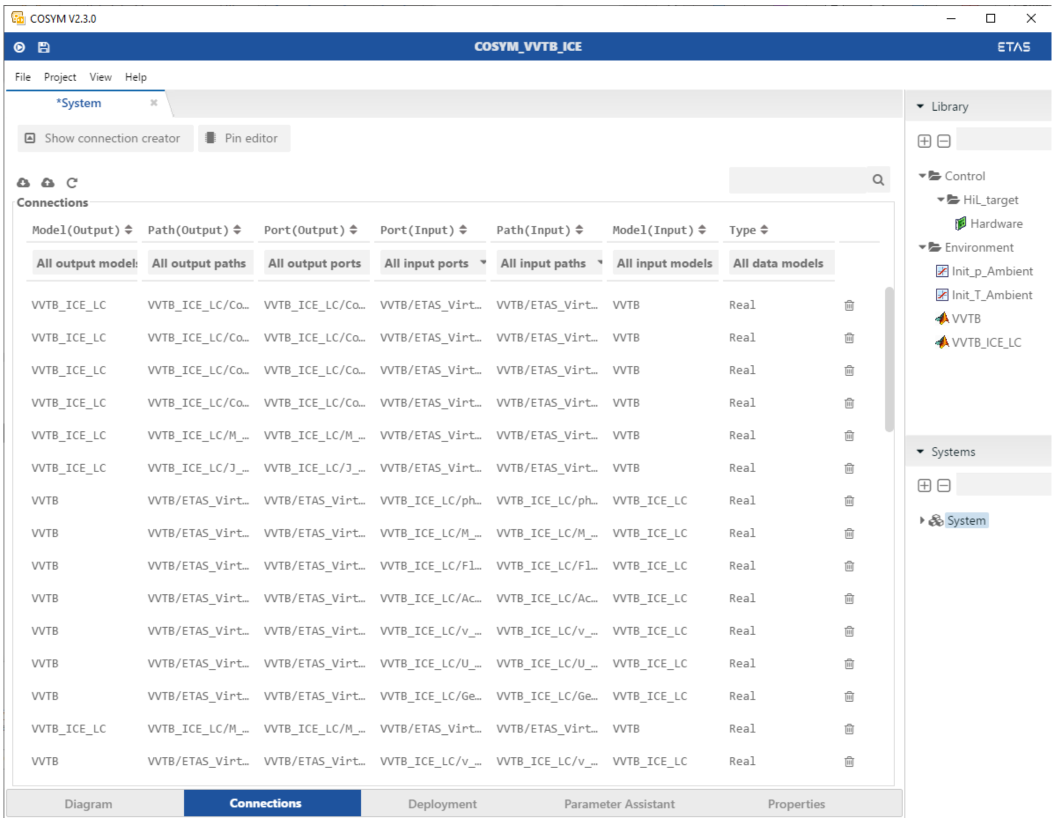
- `CU_EMS-ICE.xml`
- `VVTB-Init_p_Ambient-ICE.xml`
- `VVTB-Init_T_Ambient-ICE.xml`

  These files are located in installation folder at
  `.\ICE\Project Items\COSYM\ICE_G_1B_1Stg_DEMO\conn`

<u>To add connection files</u>

1. Open the "Connections" view available at the bottom of the opened "System".
2. Click on the "Import connections" icon.
3. Select each XML file mentioned above.
4. Click on "Import" to import connections from each file.
5. Repeat the steps 1 to 4 for all the connection files mentioned above.

> After the connection files are added, the necessary connections are established between the VVTB, the engine model, and the SoftECU. They are visible in the "Connections" view.



Model(Output)	Path(Output)	Port(Output)	Port(Input)	Path(Input)	Model(Input)	Type
All output model:	All output paths	All output ports	All input ports	All input paths	All input models	All data models
VVTB_ICE_LC	VVTB_ICE_LC/Co...	VVTB_ICE_LC/Co...	VVTB/ETAS_Virt...	VVTB/ETAS_Virt...	VVTB	Real
VVTB_ICE_LC	VVTB_ICE_LC/Co...	VVTB_ICE_LC/Co...	VVTB/ETAS_Virt...	VVTB/ETAS_Virt...	VVTB	Real
VVTB_ICE_LC	VVTB_ICE_LC/Co...	VVTB_ICE_LC/Co...	VVTB/ETAS_Virt...	VVTB/ETAS_Virt...	VVTB	Real
VVTB_ICE_LC	VVTB_ICE_LC/Co...	VVTB_ICE_LC/Co...	VVTB/ETAS_Virt...	VVTB/ETAS_Virt...	VVTB	Real
VVTB_ICE_LC	VVTB_ICE_LC/M_...	VVTB_ICE_LC/M_...	VVTB/ETAS_Virt...	VVTB/ETAS_Virt...	VVTB	Real
VVTB_ICE_LC	VVTB_ICE_LC/J_...	VVTB_ICE_LC/J_...	VVTB/ETAS_Virt...	VVTB/ETAS_Virt...	VVTB	Real
VVTB	VVTB/ETAS_Virt...	VVTB/ETAS_Virt...	VVTB_ICE_LC/ph...	VVTB_ICE_LC/ph...	VVTB_ICE_LC	Real
VVTB	VVTB/ETAS_Virt...	VVTB/ETAS_Virt...	VVTB_ICE_LC/M_...	VVTB_ICE_LC/M_...	VVTB_ICE_LC	Real
VVTB	VVTB/ETAS_Virt...	VVTB/ETAS_Virt...	VVTB_ICE_LC/Fl...	VVTB_ICE_LC/Fl...	VVTB_ICE_LC	Real
VVTB	VVTB/ETAS_Virt...	VVTB/ETAS_Virt...	VVTB_ICE_LC/Ac...	VVTB_ICE_LC/Ac...	VVTB_ICE_LC	Real
VVTB	VVTB/ETAS_Virt...	VVTB/ETAS_Virt...	VVTB_ICE_LC/v_...	VVTB_ICE_LC/v_...	VVTB_ICE_LC	Real
VVTB	VVTB/ETAS_Virt...	VVTB/ETAS_Virt...	VVTB_ICE_LC/U_...	VVTB_ICE_LC/U_...	VVTB_ICE_LC	Real
VVTB	VVTB/ETAS_Virt...	VVTB/ETAS_Virt...	VVTB_ICE_LC/Ge...	VVTB_ICE_LC/Ge...	VVTB_ICE_LC	Real
VVTB_ICE_LC	VVTB_ICE_LC/M_...	VVTB_ICE_LC/M_...	VVTB/ETAS_Virt...	VVTB/ETAS_Virt...	VVTB	Real
VVTB	VVTB/ETAS_Virt...	VVTB/ETAS_Virt...	VVTB_ICE_LC/v_...	VVTB_ICE_LC/v_...	VVTB_ICE_LC	Real

Fig. 5-1 COSYM Connections View for VVTB_ICE_LC

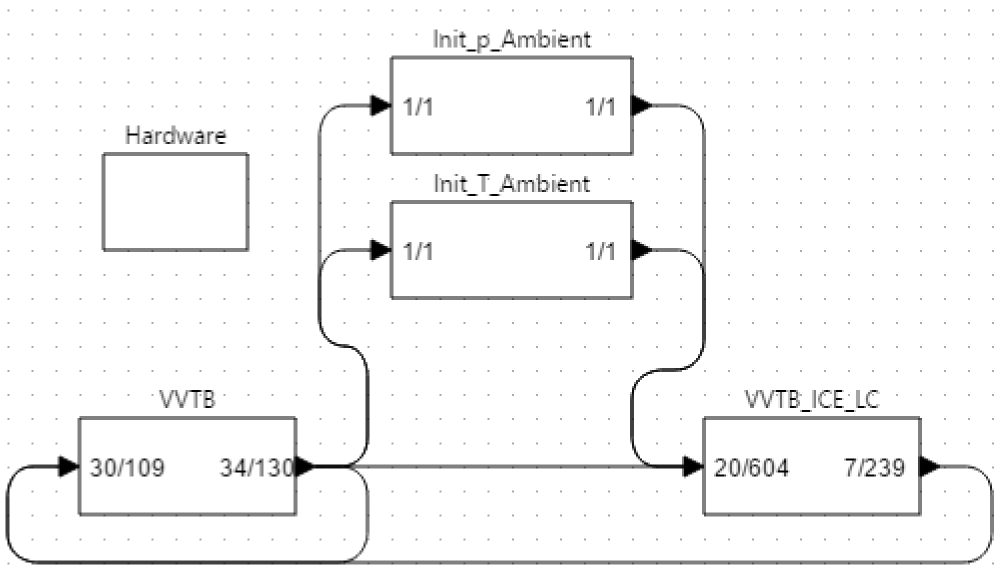


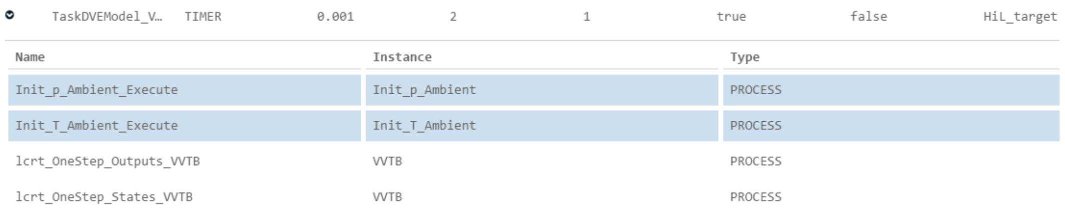
Fig. 5-2 COSYM Diagram View with Connections VVTB_ICE_LC

NOTE

Check each connection file and ensure that you added all connection files. Otherwise, the COSYM project might not work properly.

OS Configuration

Via the "Deployment" view, you can organize the processes of COSYM modules. For Simulink modules, the order of execution and the selection of a proper step size is important. By default, tasks are *not* auto-generated for Simulink modules when adding them to the project. Please click on **+ Auto mapping** to automatically map the processes to tasks. For added C-Code modules, ensure that the timer processes under tasks are executed before the Simulink model processes as shown in



```
TaskDVEModel_V…   TIMER   0.001   2   1   true   false   HiL_target
```

Name	Instance	Type
Init_p_Ambient_Execute	Init_p_Ambient	PROCESS
Init_T_Ambient_Execute	Init_T_Ambient	PROCESS
lcrt_OneStep_Outputs_VVTB	VVTB	PROCESS
lcrt_OneStep_States_VVTB	VVTB	PROCESS

Fig. 5-3 Execution of Timer Processes

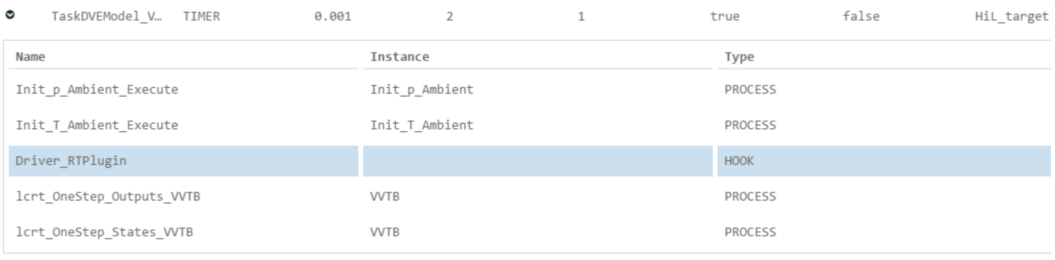
NOTE

Both "Init" processes of C-Code modules must be assigned to the "Init" task. This happens automatically when choosing the option **+ Auto mapping**.

In addition to the model tasks, it is important to generate a process hook. The RT-Plugins require such process hooks.

To add RT-Plugins:

1. Right-click on `TaskDVEModel`.
2. Choose "Add Hook".
3. Name the hook `Driver_RTPlugin`.
4. Move this hook before the process `lcrt_OneStep_Outputs_VVTB`.



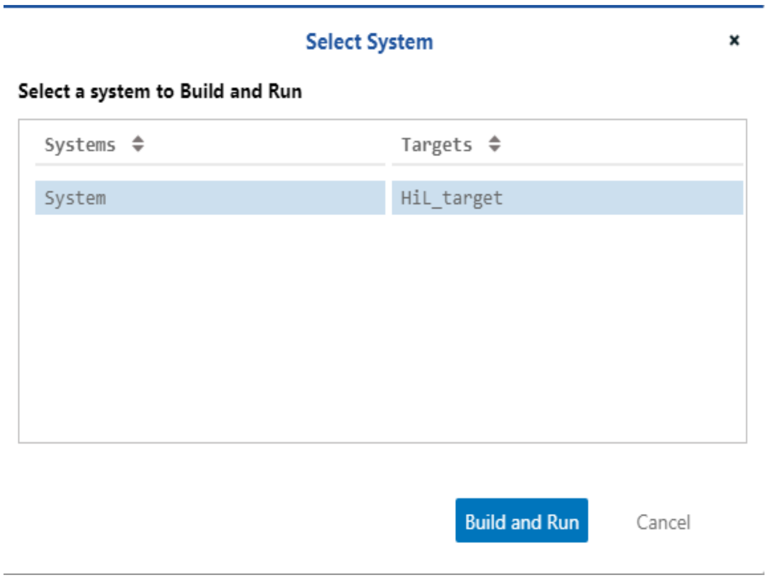
TaskDVEModel_V… TIMER 0.001 2 1 true false HiL_target

Name	Instance	Type
Init_p_Ambient_Execute	Init_p_Ambient	PROCESS
Init_T_Ambient_Execute	Init_T_Ambient	PROCESS
Driver_RTPlugin		HOOK
lcrt_OneStep_Outputs_VVTB	VVTB	PROCESS
lcrt_OneStep_States_VVTB	VVTB	PROCESS

> All relevant modules to operate the vehicle and the engine model are available and you can build a COSYM project.

To build a COSYM project

1. Choose "Project" → "Build and Run".
2. Click **Build and Run.**



Select System

Select a system to Build and Run

Systems	Targets
System	HiL_target

For more details about the Simulink code generation, refer to the section “Simulink Model - Code Generation Process” on page 14.

5.1.2 Project Creation in LABCAR-OPERATOR

This section addresses qualified personnel with knowledge in using LABCAR-OPERATOR. If you do not have any experience in using LABCAR-OPERATOR, please read first [1].

For the manual project creation using LABCAR-MODEL-ICE with LABCAR-OPERATOR follow the steps described in this section.

To set up the LABCAR-OPERATOR project

1. Open the LABCAR-IP application.
2. Click **File** → **New Project**.
3. Choose the location for the file and name the project.
4. Use "RTPC" as "Target Name".
5. Click **Finish**.

> The project explorer contains an empty hardware configuration.

The full vehicle model as well as the internal combustion engine model consist each out of a single Simulink module. This module also contains driver, environment, vehicle and SoftECU, which mimics the behavior of an EMS and will control the internal combustion engine model.

To add the virtual vehicle testbench model

1. Choose **Project** → **Add Module.**
2. Select "Add Simulink (TM) Module".
3. Select "Use existing Simulink (TM) Model (will be copied)".
4. Select the full vehicle model configuration **VVTB_MT_ICE**. The according model `VVTB.mdl` is located in the install folder `.\VVTB\Project Items\LCO\VVTB_MT_ICE\mdl\Simulink_VVTB\src`.
5. Click **Finish**.

> The Project Explorer shows the added module.

To add the internal combustion engine model

1. Choose **Project** → **Add Module.**
2. Select "Add Simulink (TM) Module".
3. Select "Use existing Simulink (TM) Model (will be copied)".
4. Select the internal combustion engine model configuration **ICE_G_1B_1Stg_DEMO**. The according model `VVTB_ICE_LC.mdl` is located in the install folder `.\ICE\Project Items\LCO\ICE_G_1B_1Stg_DEMO\mdl\Simulink_VVTB_ICE_LC\src`.
5. Activate the checkbox "Copy Model Directory into Project".
6. Click **Finish**.

> The Project Explorer shows the added model.

The internal combustion engine model requires initial environment conditions for proper simulation. For this purpose, C-Code modules are used for temperature and pressure initialization.

To add a C-Code module

1. Choose **Project** → **Add Module**.
2. Select "Add C-Code Module".
3. Select "Use existing Module (will be copied)".

4. Select the C-Code module within the internal combustion engine model configuration **ICE_G_1B_1Stg_DEMO** for ambient pressure initialization. The according module `Init_p_Ambient.lmd` is located in the install folder `.\ICE\Project Items\LCO\ICE_G_1B_1Stg_DEMO\cmod\Init_p_Ambient`.
5. Click **Finish**.

> The Project Explorer shows the added C-Code module.

To add the C-Code module for temperature initialization

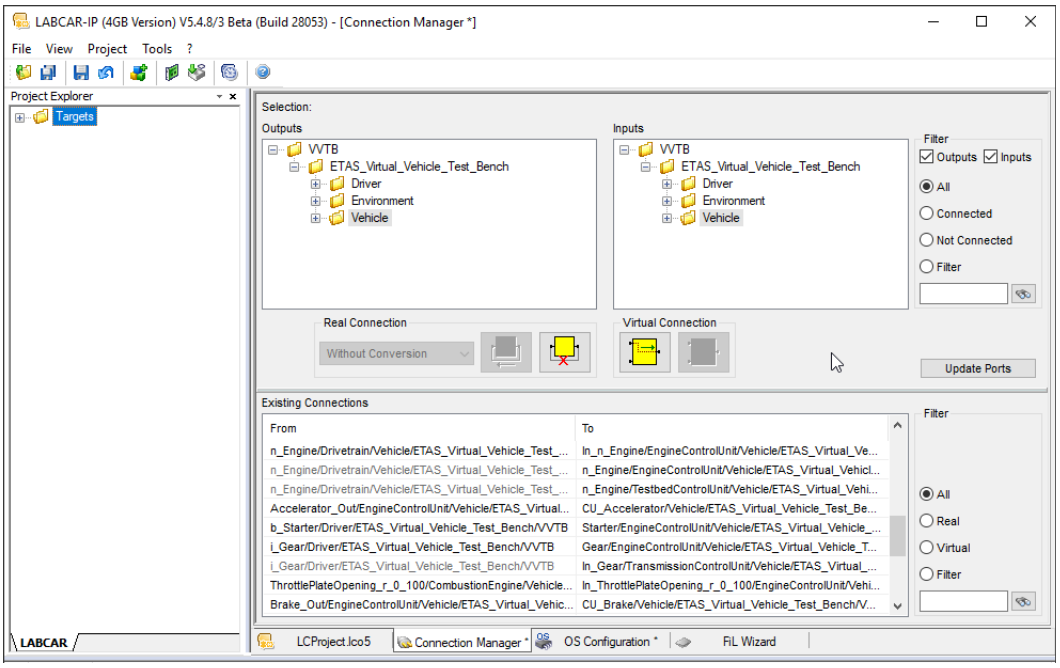
1. Choose **Project → Add Module**.
2. Select "Add C-Code Module".
3. Select "Use existing Module (will be copied)".
4. Select the C-Code module within the internal combustion engine model configuration **ICE_G_1B_1Stg_DEMO** for ambient temperature initialization. The according module `Init_T_Ambient.lmd` is located in the install folder `.\ICE\Project Items\LCO\ICE_G_1B_1Stg_DEMO\cmod\Init_T_Ambient`.
5. Click **Finish**.
6. The Project Explorer shows the added C-Code module.

Per default, no model connections are established. For example the SoftECU is not connected to the vehicle, driver model or engine model, although it is part of the already integrated Simulink module `VVTB.mdl`. Pre-defined connection files couple the vehicle model, the engine model and the SoftECU.

To add connection files

1. Open "Connection Manager" and right-click.
2. Select **Import Connection**.
3. Import each connection file stored in the install folder at `.\VVTB\Project Items\LCO\VVTB_MT_ICE\conn` and at `.\ICE\Project Items\LCO\ICE_G_1B_1Stg_DEMO\conn`.

> After the connection files are added, many connections are visible in the "Connection Manager".



From	To
n_Engine/Drivetrain/Vehicle/ETAS_Virtual_Vehicle_Test_...	In_n_Engine/EngineControlUnit/Vehicle/ETAS_Virtual_Ve...
n_Engine/Drivetrain/Vehicle/ETAS_Virtual_Vehicle_Test_...	n_Engine/EngineControlUnit/Vehicle/ETAS_Virtual_Vehicl...
n_Engine/Drivetrain/Vehicle/ETAS_Virtual_Vehicle_Test_...	n_Engine/TestbedControlUnit/Vehicle/ETAS_Virtual_Vehi...
Accelerator_Out/EngineControlUnit/Vehicle/ETAS_Virtual...	CU_Accelerator/Vehicle/ETAS_Virtual_Vehicle_Test_Be...
b_Starter/Driver/ETAS_Virtual_Vehicle_Test_Bench/VVTB	Starter/EngineControlUnit/Vehicle/ETAS_Virtual_Vehicle_...
i_Gear/Driver/ETAS_Virtual_Vehicle_Test_Bench/VVTB	Gear/EngineControlUnit/Vehicle/ETAS_Virtual_Vehicle_T...
i_Gear/Driver/ETAS_Virtual_Vehicle_Test_Bench/VVTB	In_Gear/TransmissionControlUnit/Vehicle/ETAS_Virtual_...
ThrottlePlateOpening_r_0_100/CombustionEngine/Vehicle...	In_ThrottlePlateOpening_r_0_100/EngineControlUnit/Vehi...
Brake_Out/EngineControlUnit/Vehicle/ETAS_Virtual_Vehic...	CU_Brake/Vehicle/ETAS_Virtual_Vehicle_Test_Bench/V...

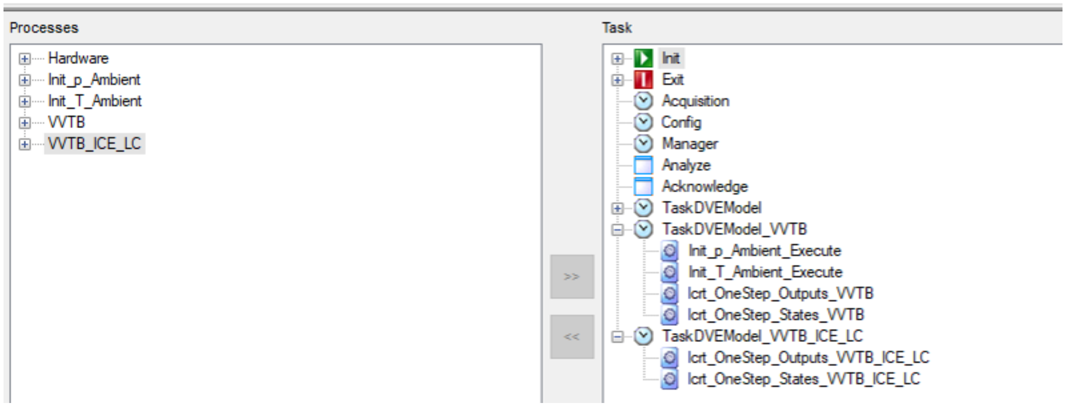
NOTE

Check each connection file and ensure that you added all connection files. Otherwise the model might not work properly.

OS Configuration

Via the "OS Configuration" tab, you can organize the processes of LABCAR-OPERATOR modules. For Simulink modules, the order of execution and the selection of a proper step size is of importance. Per default, proper tasks for the Simulink model are auto-generated when adding the module to the project. For added C-Code modules you have to assign the execute process manually.

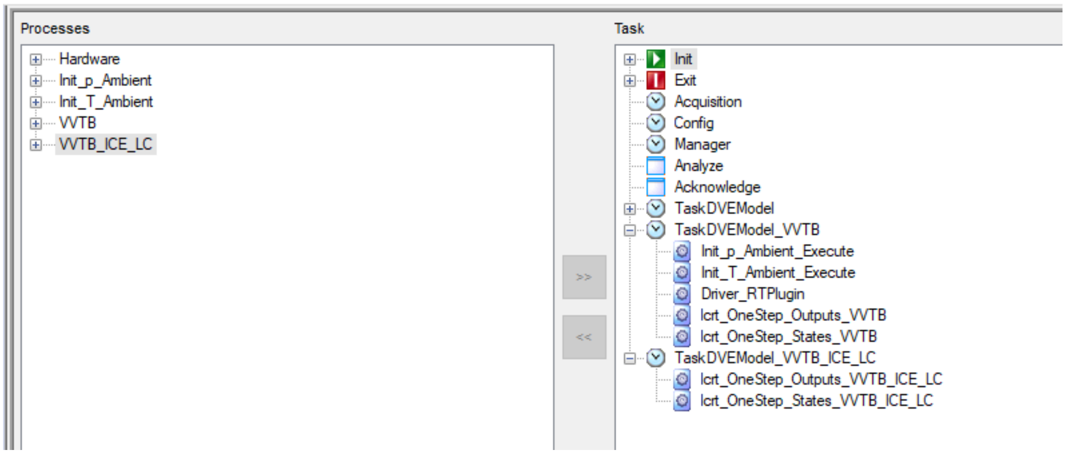
First assign the two processes `Init_p_Ambient_Execute` and `Init_T_Ambient_Execute` to the task `TaskDVEModel`.They are the first two tasks of the execution order. Please ensure that both "Init" processes are assigned to the "Init" task. In most cases this happens automatically.



In addition to the model tasks, it is important to generate a process hook. The RT-Plugins require such process hooks.

To add RT-Plugins

1. Generate a process hook for the task `TaskDVEModel`.
2. Name the hook `Driver_RTPlugin`.
3. Move this hook before the task `lcrt_OneStep_Outputs_VVTB`.



> All relevant modules to operate the vehicle and the engine model are available.

As a next step, you can build the LABCAR-OPERATOR project.

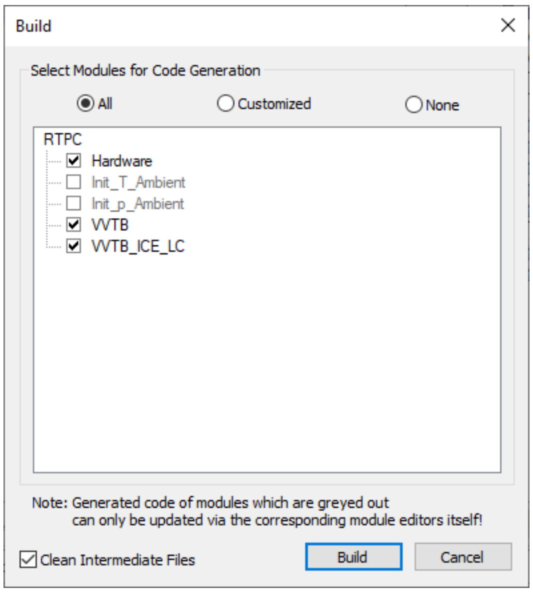
NOTE

Be sure to select the proper MATLAB version using the MATLAB Version Selector. See [2].

To build a LABCAR-OPERATOR project

1. Choose **Project** → **Build.**
2. Activate code generation for the "VVTB" module and the "VVT-B_ICE_LC" module.

3. Click **Build.**



Build

Select Modules for Code Generation

[✓ All] Customized None

RTPC
- ☑ Hardware
- ☐ Init_T_Ambient
- ☐ Init_p_Ambient
- ☑ VVTB
- ☑ VVTB_ICE_LC

Note: Generated code of modules which are greyed out can only be updated via the corresponding module editors itself!

☑ Clean Intermediate Files

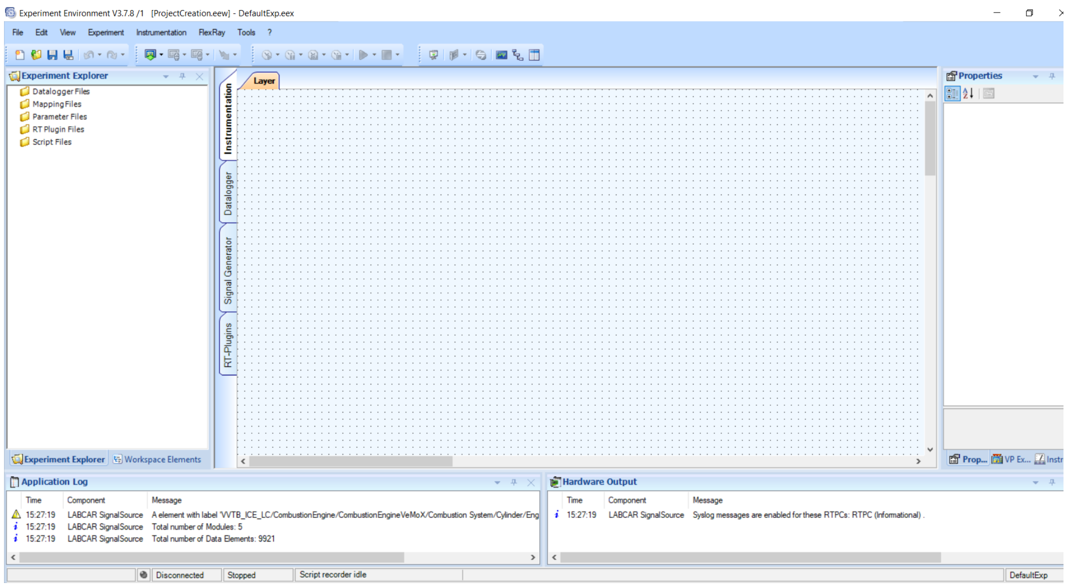
The build process performs the following basic steps:

- Simulink code generation on the host PC
- Transfer of the source code to RTPC
- Compilation of sources on RTPC
- Transfer of binaries back to the host PC

For more details about Simulink Code Generation, see “Simulink Model - Code Generation Process” on page 14.

5.1.3 Experiment Creation

After the successful build of the project in COSYM or LABCAR-OPERATOR, you can prepare the experiment for the first execution of the simulation model. Starting point is an empty experiment.



Application Log

Time	Component	Message
15:27:19	LABCAR SignalSource	A element with label 'VVTB_ICE_LC/CombustionEngine/CombustionEngineVeMoX/Combustion System/Cylinder/Eng
15:27:19	LABCAR SignalSource	Total number of Modules: 5
15:27:19	LABCAR SignalSource	Total number of Data Elements: 9921

Hardware Output

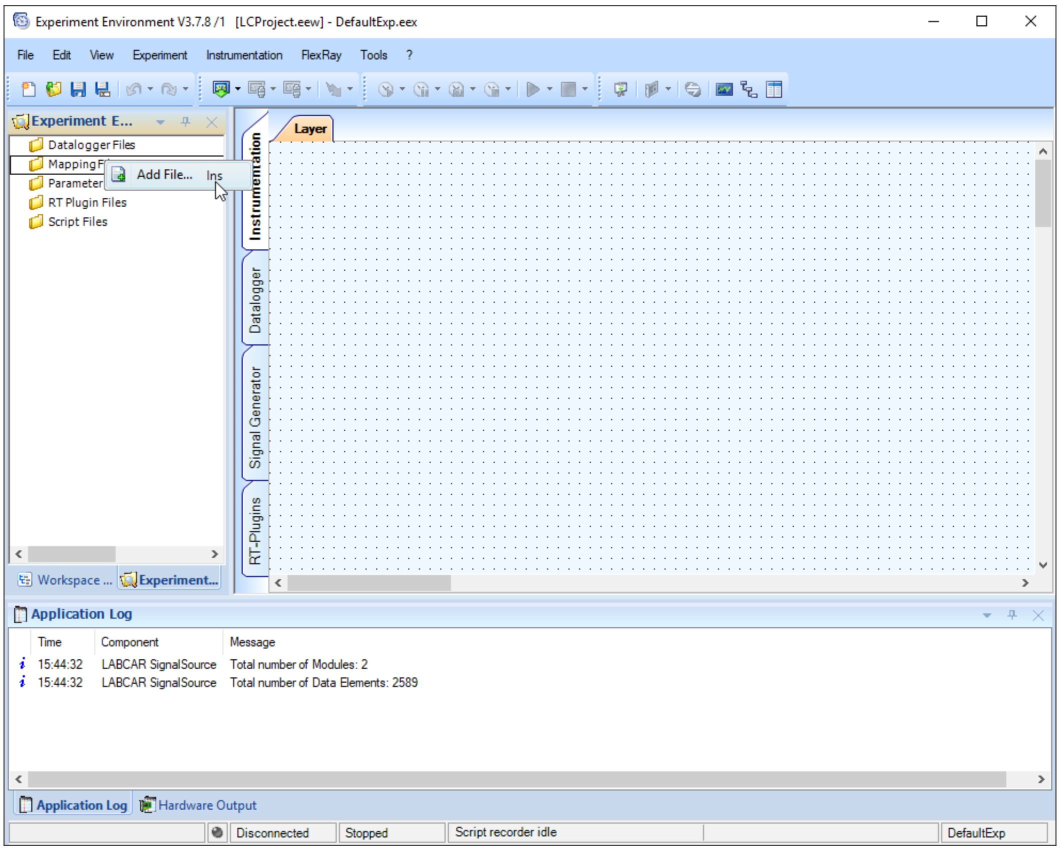
Time	Component	Message
15:27:19	LABCAR SignalSource	Syslog messages are enabled for these RTPCs: RTPC (Informational) .

Disconnected | Stopped | Script recorder idle | DefaultExp

Mapping Files

"Mapping files" provide a possibility to simplify the signal handling in LABCAR-EE. For LABCAR-MODEL-VVTB various signal mappings are pre-defined. This allows a simple access to most important simulation signals without in-depth knowledge of the model.

1. Open **Experiment Explorer**
2. Add all mapping files located in the install folder at `.\VVTB\Project Items\LCO\VVTB_MT_ICE\mapping`.



Time	Component	Message
15:44:32	LABCAR SignalSource	Total number of Modules: 2
15:44:32	LABCAR SignalSource	Total number of Data Elements: 2589

NOTE

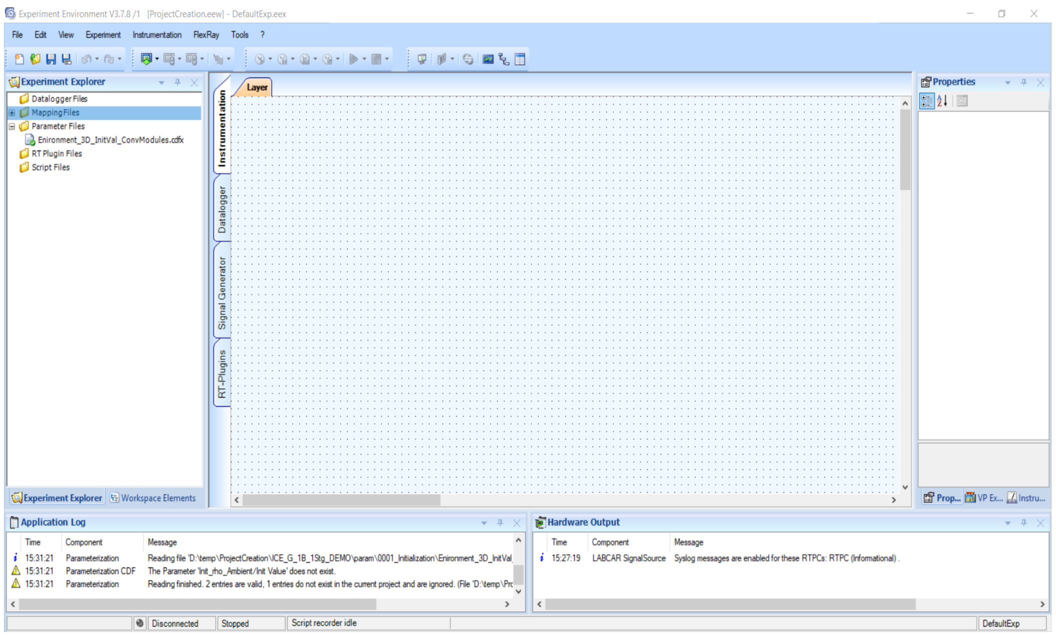
Ensure that the added mapping files are also activated.

Parameter Files

For LABCAR-MODEL-VVTB, no parameter files are provided. The default values of the Simulink model are used.

For LABCAR-MODEL-ICE, a parameter file is provided to set the ambient environment conditions.

- Open **Experiment Explorer**
- Add all cdfx files located in the install folder at `.\ICE\Project Items\LCO\ICE_G_1B_1Stg_DEMO\param\0001_Initialization`.



Application Log

```
  Time      Component                 Message
i 15:31:21  Parameterization          Reading file 'D:\temp\ProjectCreation\ICE_G_1B_1Stg_DEMO\param\0001_Initialization\Enironment_3D_InitVal_
⚠ 15:31:21  Parameterization CDF      The Parameter 'Init_rho_Ambient/Init Value' does not exist.
⚠ 15:31:21  Parameterization          Reading finished. 2 entries are valid, 1 entries do not exist in the current project and are ignored. (File 'D:\temp\Pro
```

Hardware Output

```
  Time      Component                 Message
i 15:27:19  LABCAR SignalSource       Syslog messages are enabled for these RTPCs: RTPC (Informational) .
```

NOTE

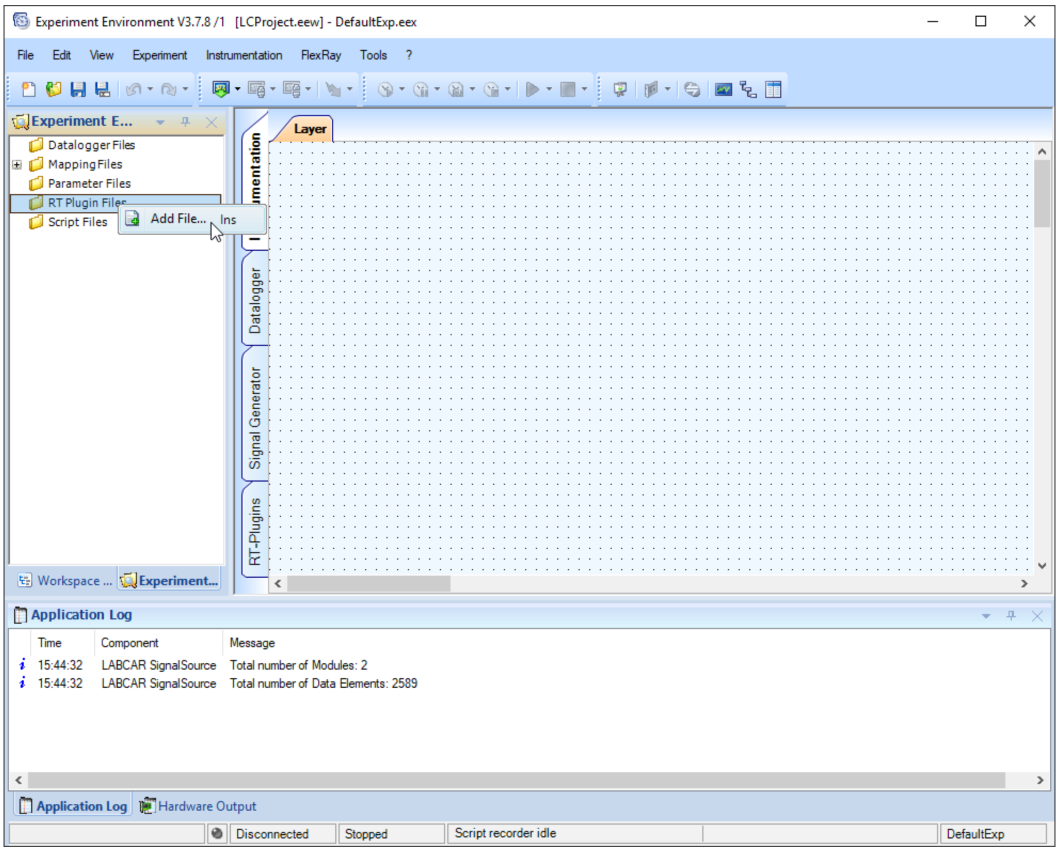
Ensure that the added parameter files are also activated.

RT-Plugin

RT-Plugins are compiled c-code snippets, which can be added to an experiment without rebuilding the complete project. It is a tool to add user-defined functionality to an experiment. The GUI of the example project uses a RT-Plugin to extend the functionality of the default instruments of LABCAR-EE.

To add RT-Plugins

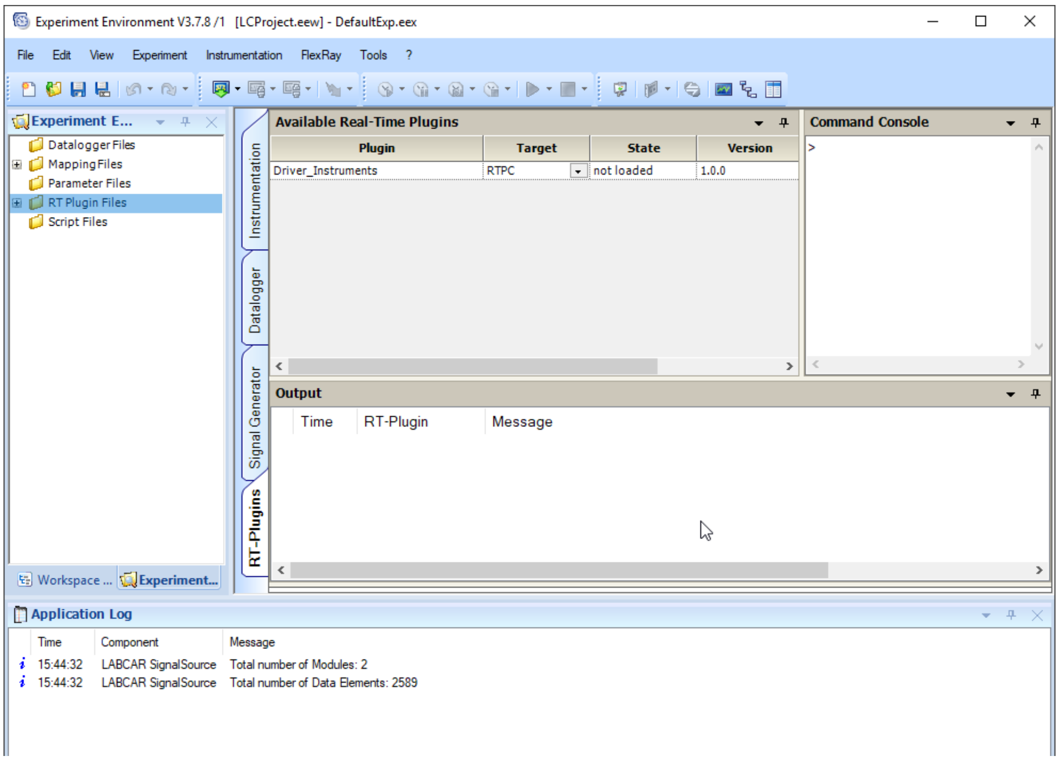
1. Open **Experiment Explorer**
2. Add RT-Plugins located in the install folder at `.\VVTB\Project Items\LCO\VVTB_MT\rtp\Driver_Instruments`.



```
Time      Component              Message
15:44:32  LABCAR SignalSource    Total number of Modules: 2
15:44:32  LABCAR SignalSource    Total number of Data Elements: 2589
```

> After the RT-Plugin is added, it is also available in the RT-Plugin tab of LABCAR-EE. Open it to check its availability.

3. Open **RT-Plugins** tab.



Plugin	Target	State	Version
Driver_Instruments	RTPC	not loaded	1.0.0

```
Time    RT-Plugin    Message
```

```
Time      Component              Message
15:44:32  LABCAR SignalSource    Total number of Modules: 2
15:44:32  LABCAR SignalSource    Total number of Data Elements: 2589
```

NOTE

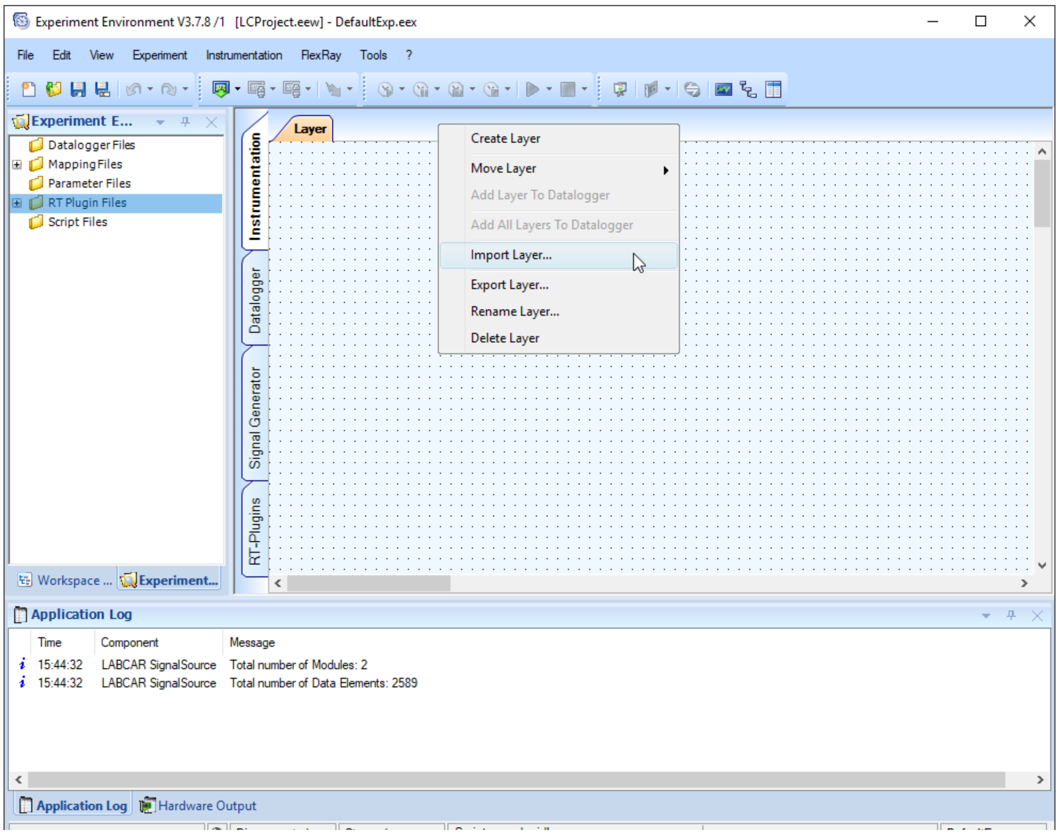
Each RT-Plugin requires a process hook for execution. Ensure that the process hook "Driver_RTPlugin" is available in the OS-settings. See the paragraph "To add RT-Plugins" on page 35.

Layers

The "Layer" provides the basic interface to handle the vehicle model. It allows the setting of the driver mode and the selection of pre-defined cycles. As a preliminary step, you can copy images for the layer from the installation folder.

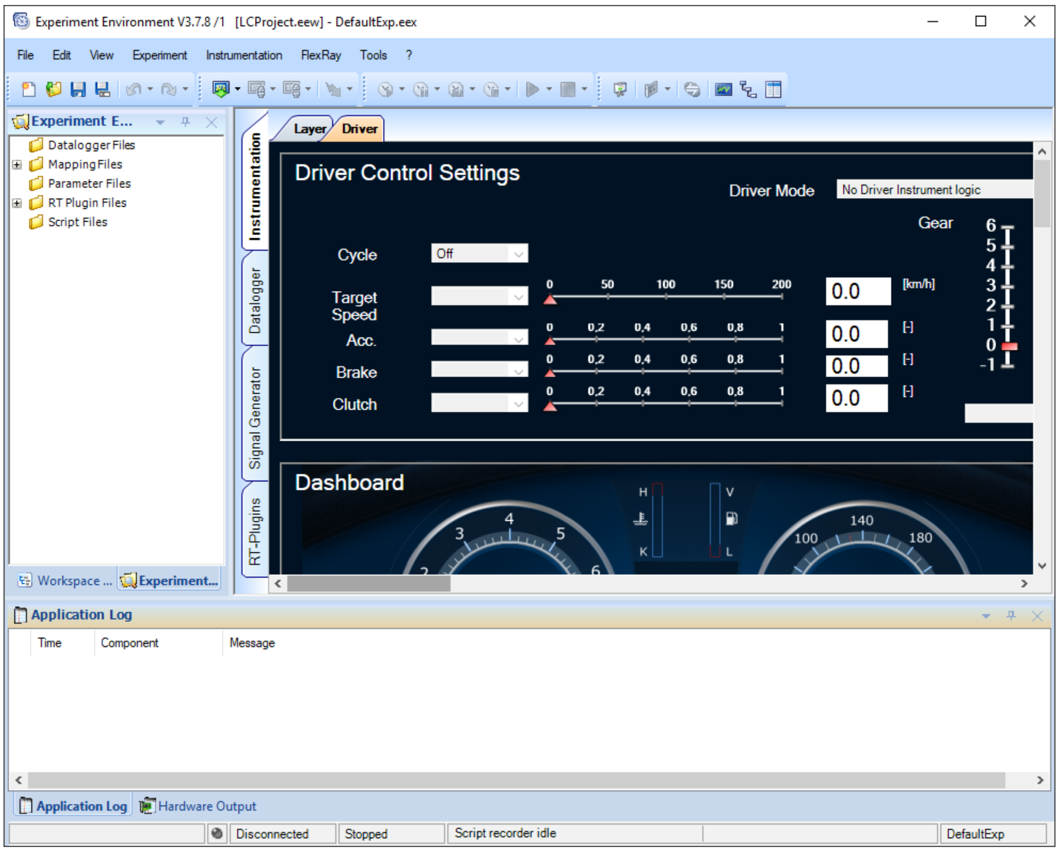
To import layers

1. Copy images from `.\VVTB\Project Items\LCO\VVTB_MT_ICE\img`.
2. Create the target folder if it does not already exist in the Experimentation folder and name it as "Image_Files".
3. Paste the images copied in step 1.
4. Import the Driver.layer file as located in `.\VVTB\Project Items\LCO\VVTB_MT_ICE\gui`.



```
   Time      Component              Message
i  15:44:32  LABCAR SignalSource    Total number of Modules: 2
i  15:44:32  LABCAR SignalSource    Total number of Data Elements: 2589
```

> After the successful import of the Driver layer, basic dashboards for handling the vehicle model exist.



NOTE

All pre-defined instruments use mapped signals provided by the mapping files. For proper execution, ensure that all mapping files are added to the experiment.

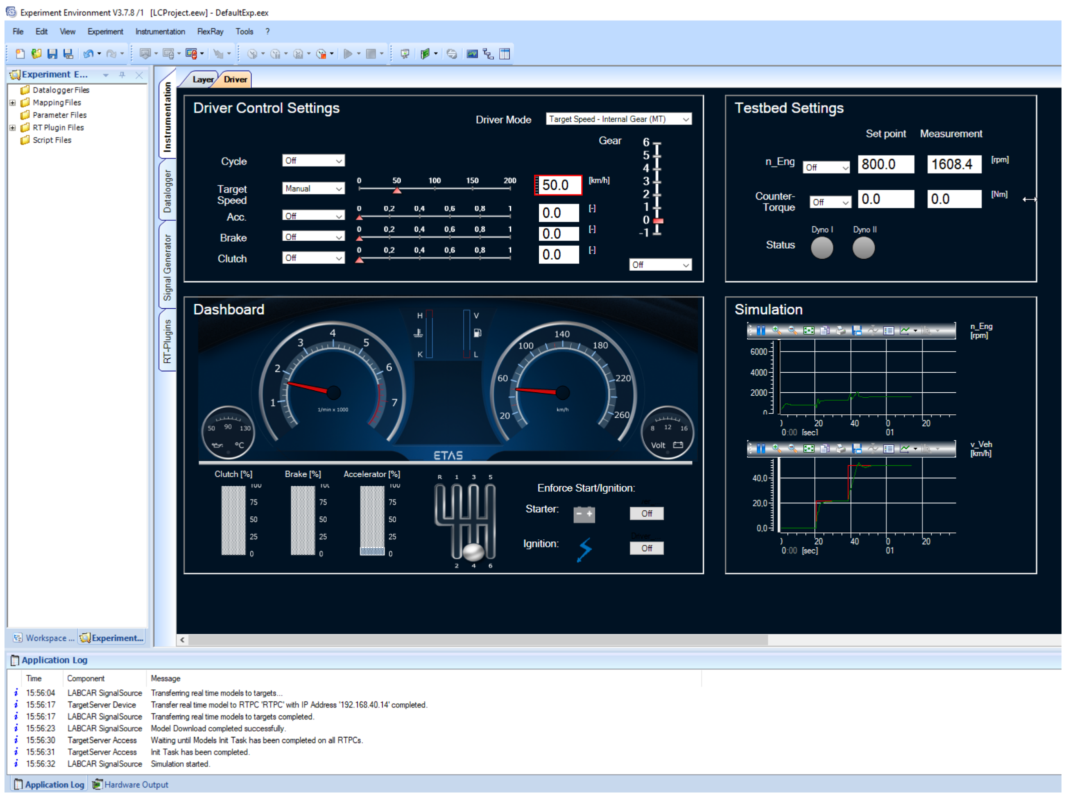
NOTE

To extend the functionality of the standard instruments, so called RT-Plugins are used. Ensure that the pre-compiled RT-Plugins are added to the experiment.

NOTE

Consider that only default LABCAR-EE instruments are used. You can adapt the layer individually to your needs.

All required artifacts for the first execution of the models are added to the experiment. With that, the experiment is ready for the download to the RTPC and the simulation. The section “Getting Started with the Tutorial Project” on page 79 describes the handling of the simulation model in more detail.



```
Time      Component            Message
15:56:04  LABCAR SignalSource  Transferring real time models to targets...
15:56:17  TargetServer Device  Transfer real time model to RTPC 'RTPC' with IP Address '192.168.40.14' completed.
15:56:17  LABCAR SignalSource  Transferring real time models to targets completed.
15:56:23  LABCAR SignalSource  Model Download completed successfully.
15:56:30  TargetServer Access  Waiting until Models Init Task has been completed on all RTPCs.
15:56:31  TargetServer Access  Init Task has been completed.
15:56:32  LABCAR SignalSource  Simulation started.
```

5.2 Getting Started with the Tutorial Project

The tutorial project for LABCAR-MODEL-ICE highlights two common aspects when using LABCAR-MODEL-ICE in a project:

- Coupling of LABCAR-MODEL-ICE with a vehicle model
- Operation of LABCAR-MODEL-ICE without a physical engine control unit

Hence, following tutorial gives an introduction on how to use LABCAR-MODEL-ICE in combination with a vehicle model. For that purpose, the detailed combustion engine variant G_1B_1Stg is coupled with a manual transmission vehicle model VVTB_MT_ICE. Please consider that the included vehicle model is part of the LABCAR-MODEL-VVTB product. To run the tutorial project successfully, a license for LABCAR-MODEL-VVTB is required and needs to be purchased separately. A physical control unit is not required, since the SoftECU of LABCAR-MODEL-VVTB will be used to control the detailed combustion engine. The tutorial project is located in
`.\Example Projectt\LCO\05_VVTB_ICE`.

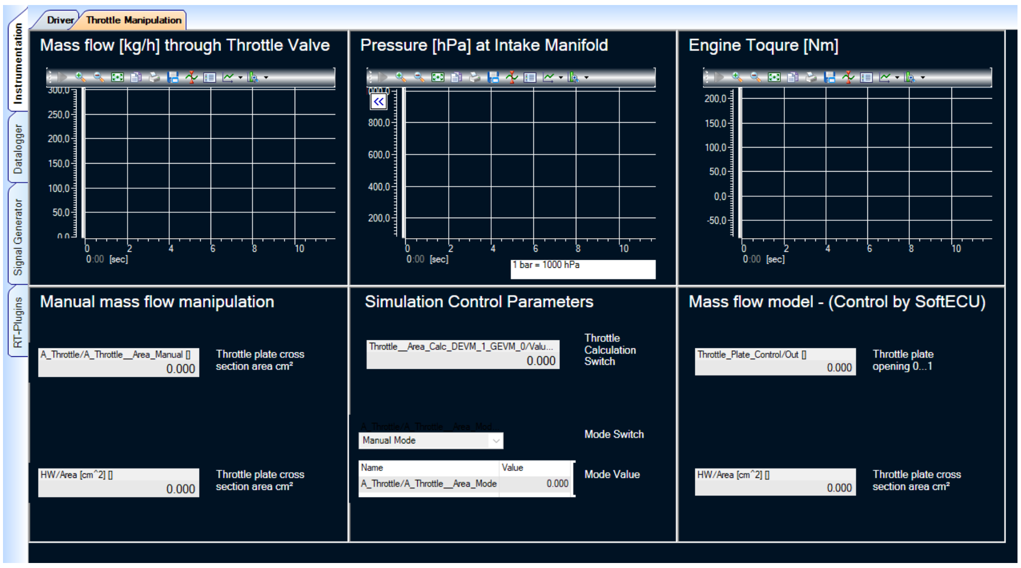
NOTE

Please do not open or manipulate the example project in the installation folder directly. Instead, copy the example project from the installation folder to a location of your choice.

5.2.1 Overview

By default, the experiment provides a single layer, which describes the driver of the vehicle model. Since this comes originally with the LABCAR-MODEL-VVTB product, see the section "User Interface" on page 52 for further information. For the following tutorial tasks, please import an additional layer to the experiment:

- Import the Throttle Manipulation layer file as located in the install folder in `.\Project Items\LCO\ICE_G_1B_1Stg_DEMO\gui`.


Driver
Throttle Manipulation
Instrumentation
Datalogger
Signal Generator
RT-Plugins
Mass flow [kg/h] through Throttle Valve
Pressure [hPa] at Intake Manifold
Engine Toqure [Nm]
[sec]
1 bar = 1000 hPa
Manual mass flow manipulation
Simulation Control Parameters
Mass flow model - (Control by SoftECU)

A_Throttle/A_Throttle__Area_Manual [] 0.000 — Throttle plate cross section area cm²

HW/Area [cm^2] [] 0.000 — Throttle plate cross section area cm²

Throttle__Area_Calc_DEVM_1_GEVM_0/Valu... 0.000 — Throttle Calculation Switch

Manual Mode — Mode Switch

Name	Value
A_Throttle/A_Throttle__Area_Mode	0.000

Mode Value

Throttle_Plate_Control/Out [] 0.000 — Throttle plate opening 0...1

HW/Area [cm^2] [] 0.000 — Throttle plate cross section area cm²

The corresponding layer provides the interface to a few characteristics of the engine model. It allows the setting of the simulation mode, the setting of the cross sectional area of the throttle plate that directly affects the air mass flow into the cylinder, and the visualization of the cross sectional area when the SoftECU controls the engine.

Oscilloscopes

The layer is divided into visualization and editing fields. The three oscilloscopes on the top of the layer show the air mass flow through the throttle, the pressure in the intake manifold before the cylinder and the generated engine torque as a main characteristic of an engine.

Parameters

The three fields at the bottom of the layer are used to switch between manual mode and model mode during simulation. In this context, the model mode can be considered as a closed loop operation, since the SoftECU replaces the functionality of the throttle plate control of a real ECU.

5.2.2 Model Control Modes

As introduced, the LABCAR-MODEL-ICE provides two model control modes:

Manual Mode

Many actuators and components provide parameters to switch and stimulate inports or outports manually. This feature allows you to operate the component in constant conditions. You can activate the mode and set a suitable value for two parameters.

Model Mode

The same actuators and components as mentioned in the section "manual mode" provide parameters to switch back to the model mode. Within this mode the engine or, more precisely, the components will be controlled by an ECU and will appear in dynamic conditions.

5.2.3 Manipulation of Air Mass Flow into Cylinder Through Throttle Plate

The manual operation is the suitable starting point to close the different control loops of a model. It is recommended to start with parametrization of stationary behavior and move iteratively to transient behavior.

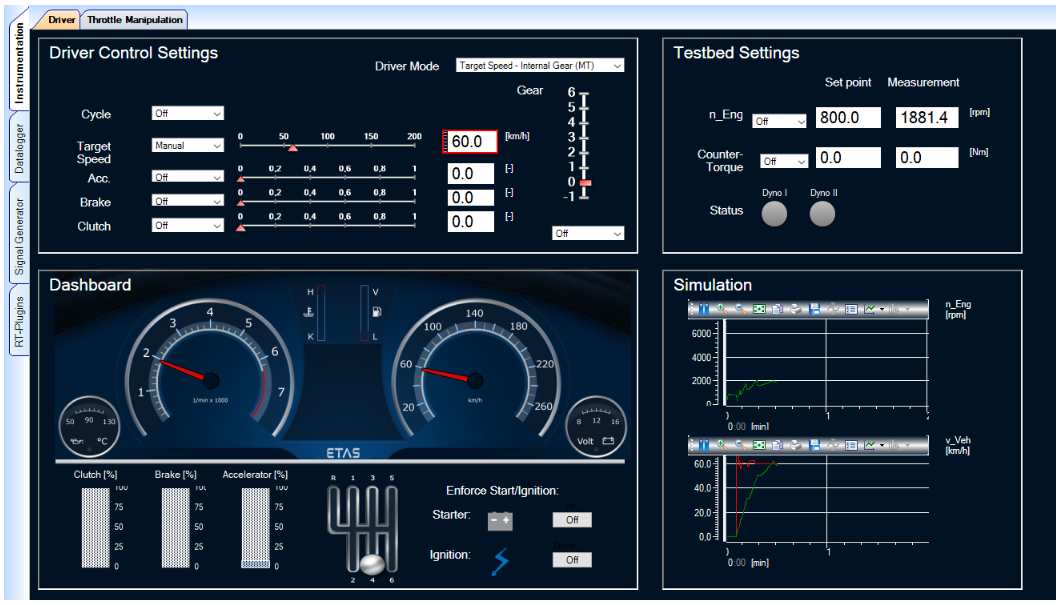
In this tutorial, the air mass flow will be directly manipulated via the cross section area of the throttle plate. By increasing the cross section area, more air mass will flow through. By decreasing the cross section area, less air is supplied to the engine for combustion.

Constant Vehicle Speed

At first, after starting the simulation, the engine will be in idle operation. To see modeled phenomena without any side effects, the vehicle will be manually accelerated until it reaches a constant vehicle speed of 60 km/h.

To set the target speed, do the following

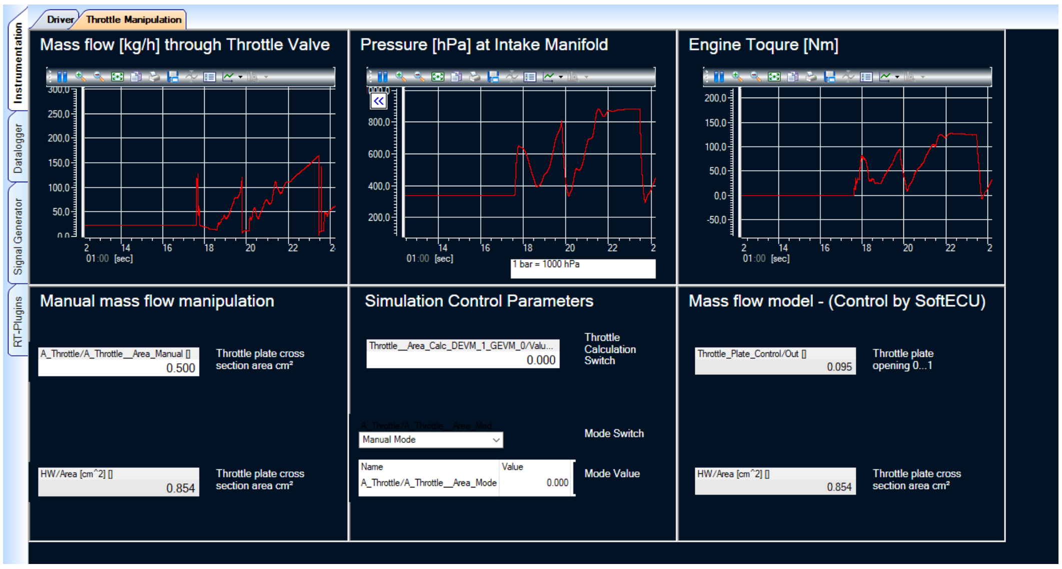
1. Switch to the drivers layer.
2. Set **Cycle** → **Off.**
3. Chose **Target Speed** → **Manual.**
4. Set the slider for vehicle speed target → **60 km/h.**



Air Mass Flow Manipulation in Model Mode

The simulation is already started in the model mode. After the vehicle reaches the desired constant velocity, switch back to the throttle manipulation layer. The oscilloscopes show a constant air mass flow, intake pressure and engine torque. As mentioned the throttle plate is controlled right from the start by the SoftECU. The parameter field "Mass flow model" visualizes one throttle plate opening in percent and the corresponding cross section if the SoftECU controls the engine.

If the engine is in idle operation, the throttle plate is almost closed. The engine has only to compensate the friction losses and no torque has to be generated. This also leads to a small air mass flow that is required to operate in idle condition. After accelerating, the engine has to generate torque. The result is that more air mass will be required for combustion and will be supplied by opening the throttle plate.


Driver
Throttle Manipulation
Mass flow [kg/h] through Throttle Valve
Pressure [hPa] at Intake Manifold
Engine Toqure [Nm]
1 bar = 1000 hPa
Manual mass flow manipulation
Simulation Control Parameters
Mass flow model - (Control by SoftECU)

A_Throttle/A_Throttle__Area_Manual [] 0.500 Throttle plate cross section area cm²

HW/Area [cm^2] [] 0.854 Throttle plate cross section area cm²

Throttle__Area_Calc_DEVM_1_GEVM_0/Valu... 0.000 Throttle Calculation Switch

Manual Mode — Mode Switch

Name: A_Throttle/A_Throttle__Area_Mode, Value: 0.000 — Mode Value

Throttle_Plate_Control/Out [] 0.095 Throttle plate opening 0...1

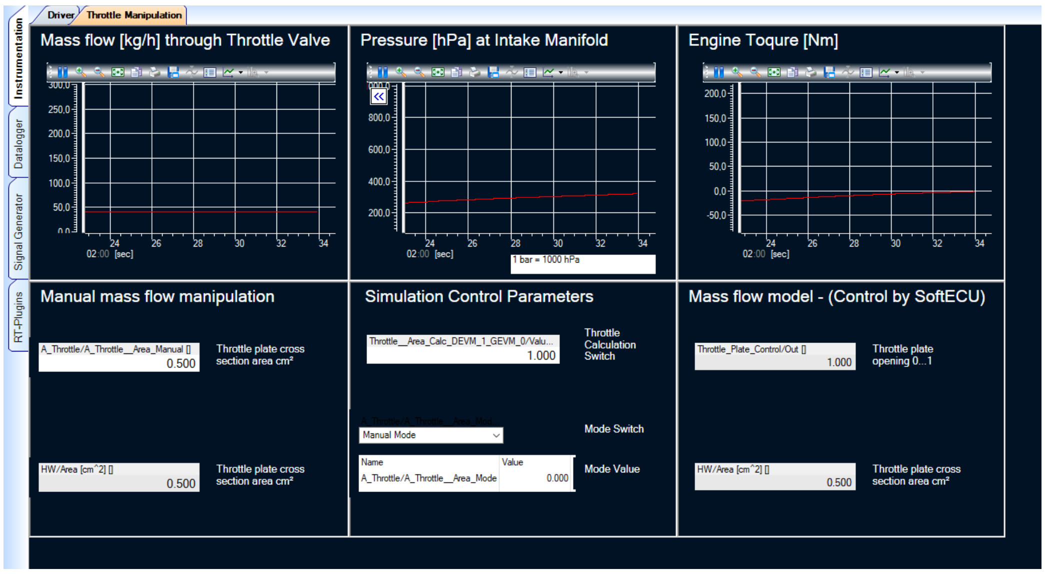
HW/Area [cm^2] [] 0.854 Throttle plate cross section area cm²

Air Mass Flow Manipulation in Manual Mode

<u>To activate the manual mode</u>

1. In the field "Simulation Control Parameters", set the parameter **Throttle__Area_Calc_DEVM_1_GEVM_0/Value** → **1**
2. Set the parameter **A_Throttle__Area_Mode** → **Manual Mode**
3. Set the parameter **A_Throttle__Area_Manual** → **0.5**

This setting allows you to directly enter the cross section area of the throttle plate. Immediately after entering the air mass flow, the pressure and the engine torque will reflect the chosen setting.



Manual mass flow manipulation

A_Throttle/A_Throttle__Area_Manual [] 0.500 — Throttle plate cross section area cm²

HW/Area [cm^2] [] 0.500 — Throttle plate cross section area cm²

Simulation Control Parameters

Throttle__Area_Calc_DEVM_1_GEVM_0/Valu... 1.000 — Throttle Calculation Switch

Manual Mode — Mode Switch

Name	Value
A_Throttle/A_Throttle__Area_Mode	0.000

Mode Value

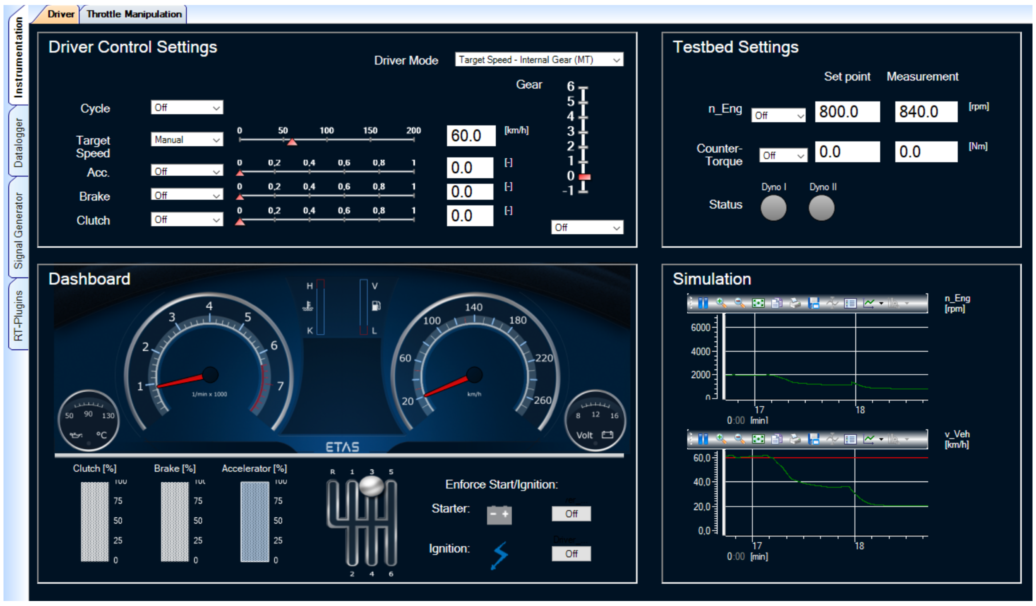
Mass flow model - (Control by SoftECU)

Throttle_Plate_Control/Out [] 1.000 — Throttle plate opening 0...1

HW/Area [cm^2] [] 0.500 — Throttle plate cross section area cm²

The following physical phenomena can be observed while manually manipulating the air mass flow:

By decreasing the cross section almost to zero, the throttle is closing. This leads to an increased fluid resistance for the air, which wants to flow into the cylinder. The engine power or/and the engine torque decreases significantly. The drivers layer visualizes the described behavior.



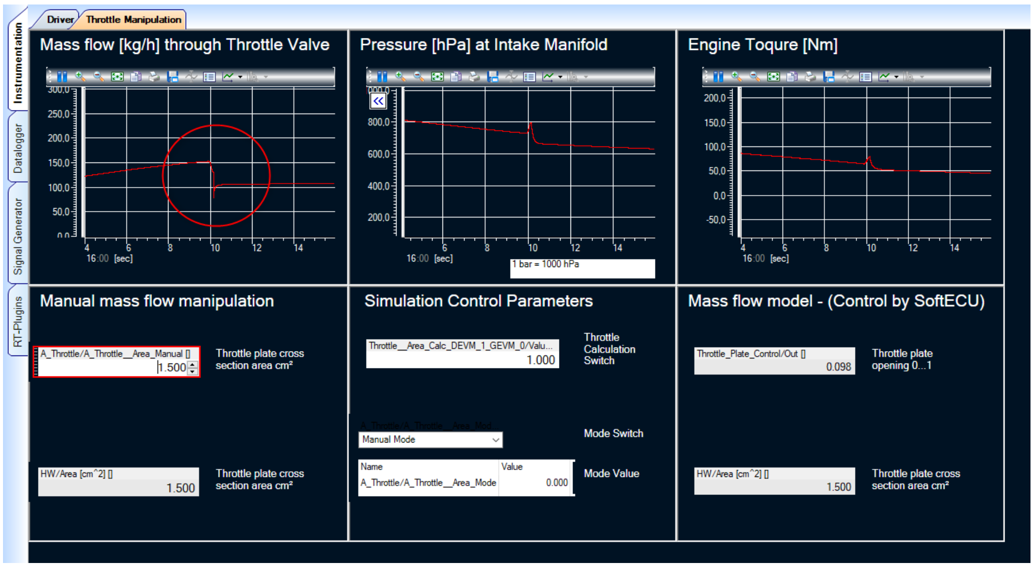
Driver Mode: Target Speed - Internal Gear (MT)
Cycle: Off
Target Speed: Manual 60.0 [km/h]
Acc.: Off 0.0 [-]
Brake: Off 0.0 [-]
Clutch: Off 0.0 [-]
Gear: Off

	Set point	Measurement	
n_Eng Off	800.0	840.0	[rpm]
Counter-Torque Off	0.0	0.0	[Nm]

Status: Dyno I, Dyno II

Enforce Start/Ignition:
Starter: Off
Ignition: Off

By increasing or decreasing, the relaxation of chambers can be observed. Looking at the mass flow oscilloscope, a peak appears after entering. This peak demonstrates the relaxation where the whole system is going to balance itself after a disturbance.



Manual mass flow manipulation
A_Throttle/A_Throttle__Area_Manual [] 1.500 — Throttle plate cross section area cm²
HW/Area [cm^2] [] 1.500 — Throttle plate cross section area cm²

Simulation Control Parameters
Throttle__Area_Calc_DEVM_1_GEVM_0/Valu... 1.000 — Throttle Calculation Switch
Manual Mode — Mode Switch
Name: A_Throttle/A_Throttle__Area_Mode, Value: 0.000 — Mode Value

Mass flow model - (Control by SoftECU)
Throttle_Plate_Control/Out [] 0.098 — Throttle plate opening 0...1
HW/Area [cm^2] [] 1.500 — Throttle plate cross section area cm²

5.3 Signal Interface

The basic interface to operate LABCAR-MODEL-ICE, as used in the section “Getting Started with the Tutorial Project” on page 79, is listed in the following table. LABCAR-MODEL-VVTB is used in combination with LABCAR-MODEL-ICE. All required connections of VVTB and its components can be found within the connections files of VVTB. Since LABCAR-MODEL-ICE is considered as an add-on in the tutorial, the table contains all required connections to satisfy when running LABCAR-MODEL-ICE in combination with LABCAR-MODEL-VVTB. If using LABCAR-MODEL-ICE with other drivetrain and driver models, the required connections will differ.

NOTE

The table only contains a small excerpt from all available input and output ports of the engine model. The model reference guide of LABCAR-MODEL-ICE provides the complete documentation of the physical interfaces of each S-Function and subsystem.

5.3.1 Combustion Engine/Top Level

Inputs

Name	Type	Unit	Description
phi_CAModel	Double	CA	Crankshaft angle
FlywheelStandstill	Bool	-	Engine operation status
Ignition	Bool	-	Ignition status
p_Ambient	Double	Pa	Ambient pressure
T_Ambient	Double	K	Ambient temperature
Accelerator	Double	-	Accelerator position
n_Engine	Double	rpm	Engine speed
U_Batt	Double	V	Battery voltage
M_Engage	Double	Nm	Torque of auxiliary systems
v_Vehicle	Double	km/h	Vehicle speed
v_Distance	Double	m	Traveled distance of vehicle
Clutch	Double	-	Clutch pedal position
Gear	Int	-	Gear position

Outputs

Name	Type	Unit	Description
M_Engine	Double	Nm	Engine torque
J_Engine	Double	kgm²	Moment of inertia of engine
M_EngineStick	Double	Nm	Engine stick torque of resting engine

5.3.2 Combustion Engine/Injection

Inputs

Name	Type	Unit	Description
t_Injection1	Double	µs	Injection time cylinder 1
t_Injection2	Double	µs	Injection time cylinder 2
t_Injection3	Double	µs	Injection time cylinder 3
t_Injection4	Double	µs	Injection time cylinder 4
t_Injection5	Double	µs	Injection time cylinder 5
t_Injection6	Double	µs	Injection time cylinder 6

5.3.3 Combustion Engine/Lambda Sensor

Inputs

Name	Type	Unit	Description
Lambda	Double	-	Lambda sensor before catalyst

5.3.4 Combustion Engine/Throttle Plate

Inputs

Name	Type	Unit	Description
ADC_Cannel_0_1	Double	-	Opening value throttle plate

5.3.5 Combustion Engine/Engine Torque

Outputs

Name	Type	Unit	Description
M_EngineOut_Nm	Double	Nm	Effective drive torque
M_Ind_Nm	Double	Nm	Indicated engine torque
M_Drag_Nm	Double	Nm	Engine drag torque

6 LABCAR-MODEL-PMSM

ETAS LABCAR-MODEL-PMSM is an FPGA-based simulation model of a 3-phase permanent magnet synchronous e-motor (PMSM). The model includes a model of an inverter, of the PMSM e-motor, and of a simple drivetrain.

6.1 Overview

As an FPGA-based simulation model, LABCAR-MODEL-PMSM requires the ETAS ES5340.2-M FPGA board for execution [3]. LABCAR-MODEL-PMSM can output signals directly via the analog output channels of the ES5340.2-M board. The additional outputs of the slave board, ES5340.1-S, are currently unsupported.

Two model variants are available: LABCAR-MODEL-PMSM Basic and LABCAR-MODEL-PMSM Advanced. In addition to all the features of the Basic variant, the Advanced variant offers a few more parameters, like iron losses in the e-motor and Ohmic resistances in the inverter.

The model uses the ETAS FlexibleFPGA toolchain. Thus, many steps require following the guidelines described in the FlexibleFPGA tutorial [4].

6.2 Project and Experiment Creation

This section lists all the necessary steps for setting up a small test project with LABCAR-MODEL-PMSM using LABCAR-OPERATOR. It is divided into three subsections that reflect the main steps: the procedure to flash LABCAR-MODEL-PMSM to an ES5340.2-M board, the project creation in LABCAR-IP, and the experiment creation in LABCAR-EE.

This section addresses qualified personnel with knowledge in using LABCAR-OPERATOR. If unsure, please consider the documentation of LABCAR-OPERATOR first [1]. When using ETAS COSYM, the steps in the second section differ slightly. Please consider the COSYM User's Guide for details [3].

NOTE

Undefined behavior of the simulation model
There are two variants of this model: LABCAR-MODEL-PMSM Basic and LACAR-MODEL-PMSM Advanced. Each variant has its own programming file (*.euf) and its own interface description file (*.adf).
Mixing the programming file from one variant with the interface description file from the other variant leads to undefined behavior.
Both files of a model variant (*.euf and *.adf) belong strictly to each other; they cannot be mixed. Always use both files in combination when configuring a project.

6.2.1 Flashing Procedure

The prerequisite for using a LABCAR-OPERATOR project is flashing the model to the ES5340.2-M. Ensure that the ES5340.2-M board is inserted to the RTPC and the host PC is connected with the RTPC.

The following steps require the FlexFPGACopy tool, which is shipped together with LABCAR-MODEL-PMSM as well as with FlexibleFPGA. It is available on the product installation medium. Additionally, the steps require the ETAS HSP Update tool. It is available either as part of LABCAR-OPERATOR or via download on the ETAS Download Server.

NOTE

The HSP update tool is delivered with LABCAR-OPERATOR until version 5.4.7. Newer versions of LABCAR-OPERATOR need a separate installation of the HSP update tool. Please consider [2] to install the correct version of the HSP update tool.

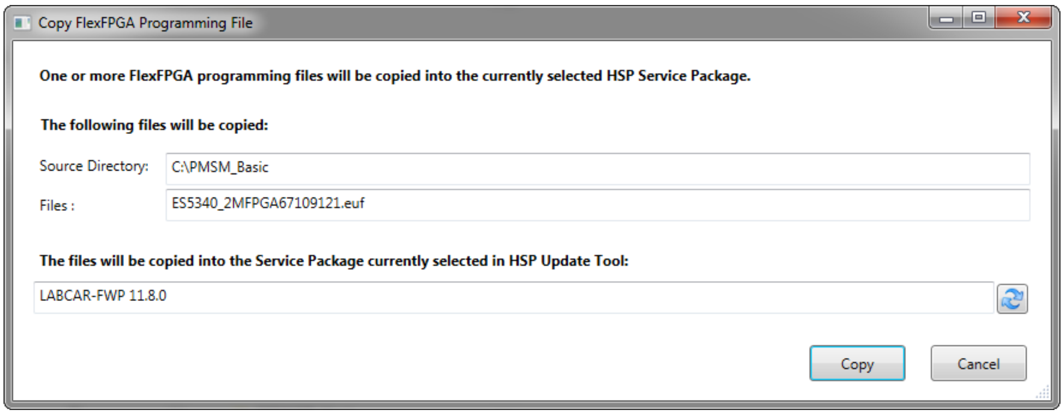
To program the LABCAR-MODEL-PMSM to the ES5340

NOTE

The HSP Update Tool flashes the model only if there is a valid FlexibleFPGA license available. For installing the license refer to [4].

1. Run "`FlexFPGACopy.exe`".

   The tool copies the PMSM model file ("`ES5340_2MFPGA67109121.euf`") to the HSP installation directory along with additional information. You find the two files "`FlexFPGACopy.exe`" and "`.euf`" in the "FPGA Models" folder of the LABCAR-MODEL-PMSM installation.
2. Click **Copy.**



One or more FlexFPGA programming files will be copied into the currently selected HSP Service Package.

The following files will be copied:

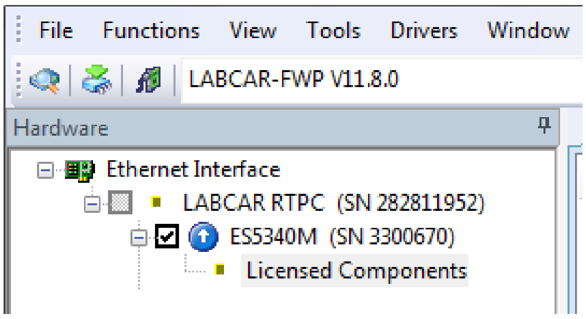
Source Directory: C:\PMSM_Basic

Files : ES5340_2MFPGA67109121.euf

The files will be copied into the Service Package currently selected in HSP Update Tool:

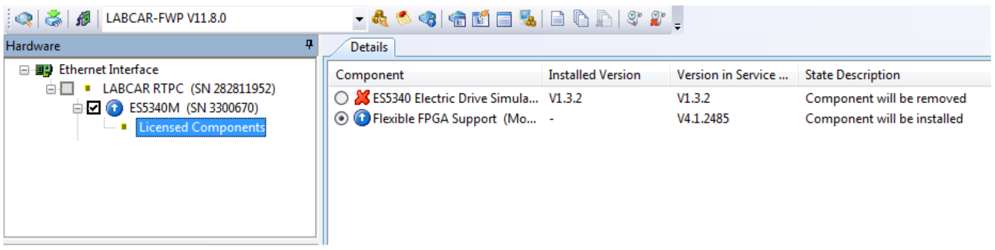
LABCAR-FWP 11.8.0

3. Start the **HSP Update Tool** and choose LABCAR-FWP service package.
4. Use Functions → Search for Hardware.



5. Select **ES5340M**.
6. Select **Licensed components**.

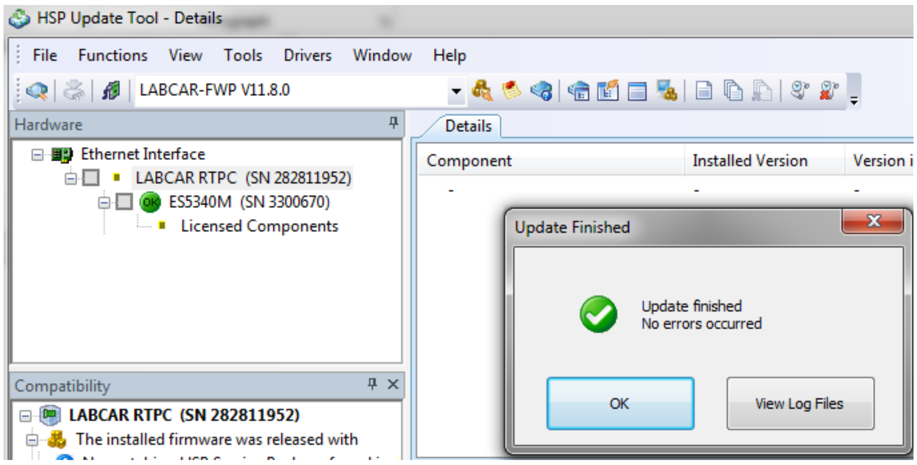
   The "Details" tab displays the licensed components of ES5340.
7. Select component **FlexibleFPGA Support**.



Component	Installed Version	Version in Service ...	State Description
ES5340 Electric Drive Simula...	V1.3.2	V1.3.2	Component will be removed
Flexible FPGA Support (Mo...	-	V4.1.2485	Component will be installed

8. Use the **Functions** → **Perform Update** option of the HSP Update Tool to flash/download the model.

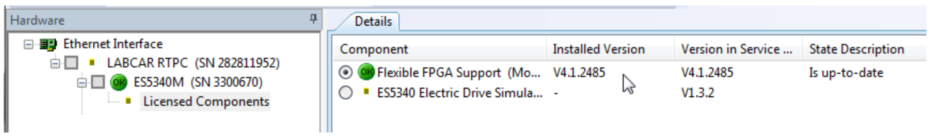
> After the successful update, the below window appears.



The installed firmware was released with

Update finished
No errors occurred

9. Click **OK**.



Component	Installed Version	Version in Service ...	State Description
Flexible FPGA Support (Mo...	V4.1.2485	V4.1.2485	Is up-to-date
ES5340 Electric Drive Simula...	-	V1.3.2	

The installed version (model version) of the "Component" "FlexibleFPGASup-port" appears in the "Details" tab.

6.2.2 Project Creation in LABCAR-OPERATOR

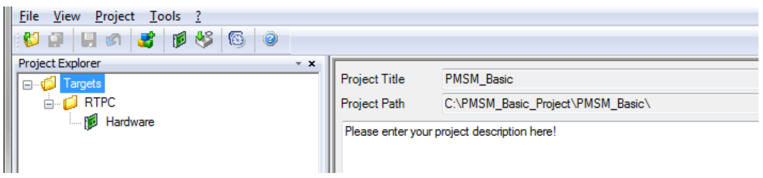
In the following, a step-by-step guide is given, which describes the generation of a LABCAR-OPERATOR project using LABCAR-MODEL-PMSM. For the execution of the project, no physical ECU is needed. Instead, inverter switches can be stimulated using failure model switch inputs. The resulting setup allows to execute the basic variant of LABCAR-MODEL-PMSM in a simple test environment.

For the manual project creation using LABCAR-MODEL-PMSM with LABCAR-OPERATOR, follow the steps described in this section.

To set up the LABCAR-OPERATOR project

1. Open LABCAR-OPERATOR.
2. Create a new project called "PMSM_Basic".

> After creation, the LABCAR-OPERATOR project environment looks as follows:

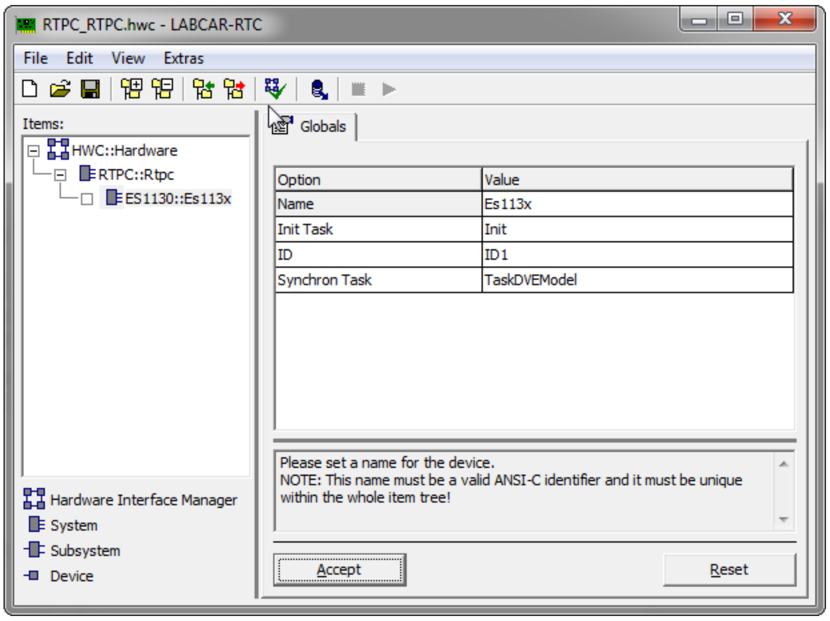


Project Title PMSM_Basic
Project Path C:\PMSM_Basic_Project\PMSM_Basic\
Please enter your project description here!

The LABCAR-MODEL-PMSM model is integrated into the LABCAR-OPERATOR project using the hardware configuration.

To integrate the LABCAR-MODEL-PMSM model

1. In the Project Explorer, right-click on "Hardware" and select **Edit.**

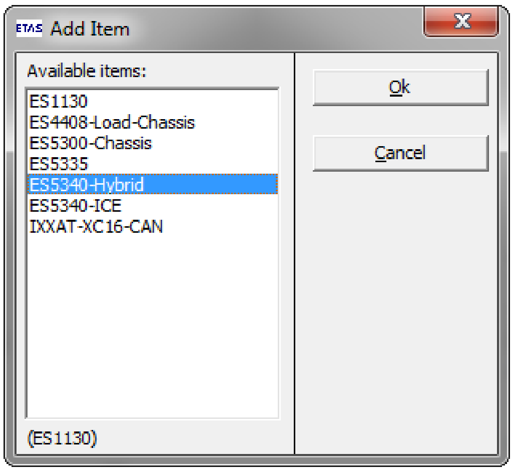


Option	Value
Name	Es113x
Init Task	Init
ID	ID1
Synchron Task	TaskDVEModel

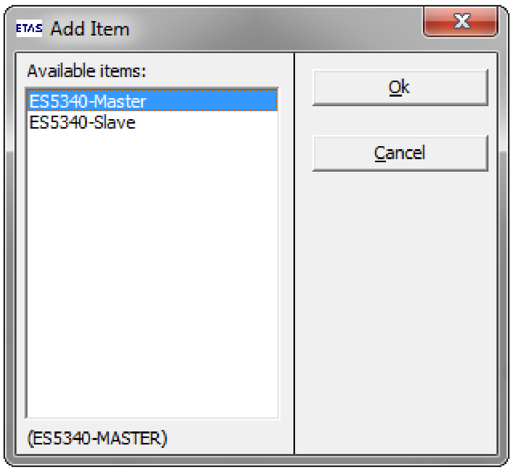
Please set a name for the device.
NOTE: This name must be a valid ANSI-C identifier and it must be unique within the whole item tree!

2. Right-click on "ES1130" and choose **Delete Item**.
3. Right-click on "RTPC::Rtpc" and choose **Add item → ES5340-Hybrid.**

4. Click **Ok.**



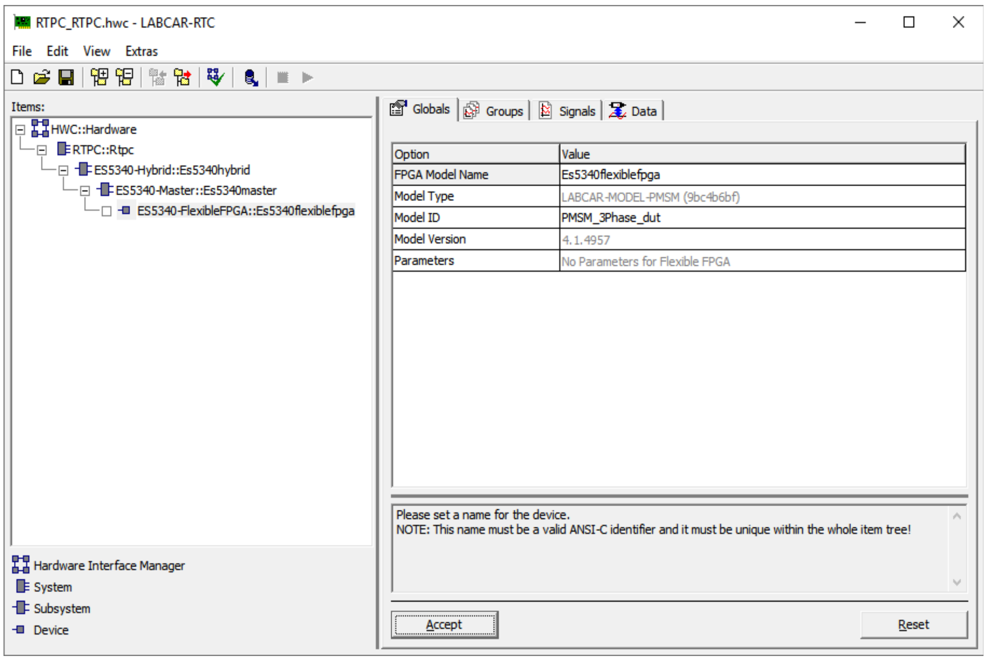
5. Right click on "ES5340-Hybrid" and choose **Add Item** → **ES5340-Master.**
6. Click **Ok.**



7. Right-click on "ES5340-Master" and choose **Add Item** → **ES5340-FlexibleFPGA.**
8. Click **Ok.**

   A window pops up.
9. Select `PMSM_3Phase_dut.adf` from `.\FPGA Models\PMSM_Basic` in the installation directory.

> Now the LABCAR-MODEL-PMSM appears on LABCAR-RTC.


RTPC_RTPC.hwc - LABCAR-RTC
File Edit View Extras
Items:
HWC::Hardware
RTPC::Rtpc
ES5340-Hybrid::Es5340hybrid
ES5340-Master::Es5340master
ES5340-FlexibleFPGA::Es5340flexiblefpga
Hardware Interface Manager
System
Subsystem
Device
Globals
Groups
Signals
Data
Accept
Reset

Option	Value
FPGA Model Name	Es5340flexiblefpga
Model Type	LABCAR-MODEL-PMSM (9bc4b6bf)
Model ID	PMSM_3Phase_dut
Model Version	4.1.4957
Parameters	No Parameters for Flexible FPGA

Please set a name for the device.
NOTE: This name must be a valid ANSI-C identifier and it must be unique within the whole item tree!

NOTE

In the "Model ID" row, the string in parentheses is used for the internal tracking reference.

The "Globals" tab contains **Model ID** and **Model Version**.

The model version appears here in "hwc" and shall match with the model version that is flashed by using the HSP tool as described in the section “Flashing Procedure” on page 89.

The "Groups" tab contains the tasks for the interfaces to the model. As default, it uses "TaskDVEModel".

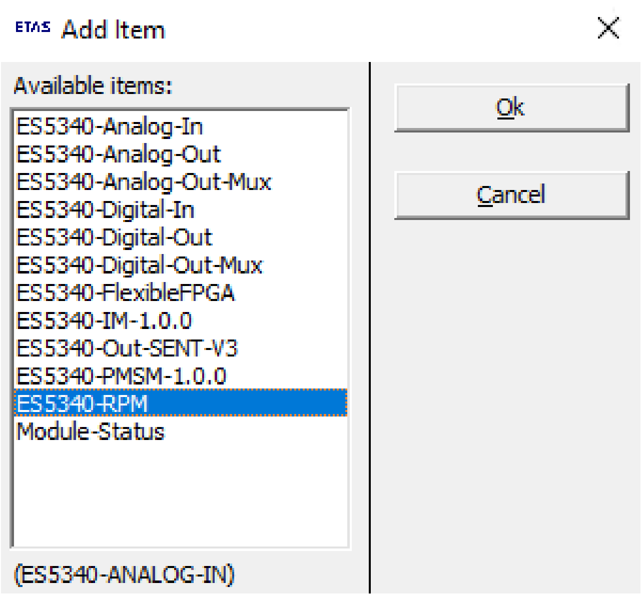
The "Signals" tab contains the signal interface of the PMSM model.

The "Data" tab contains the values of input and output signals.

In a further step, a Simulink module has to be integrated to the LABCAR-OPERATOR project to download non-linear maps to the ES5340. The section “Nonlinear Maps and Parameter Maps Downloader” on page 108 describes the generation of current maps.

<u>To integrate the RPM module</u>

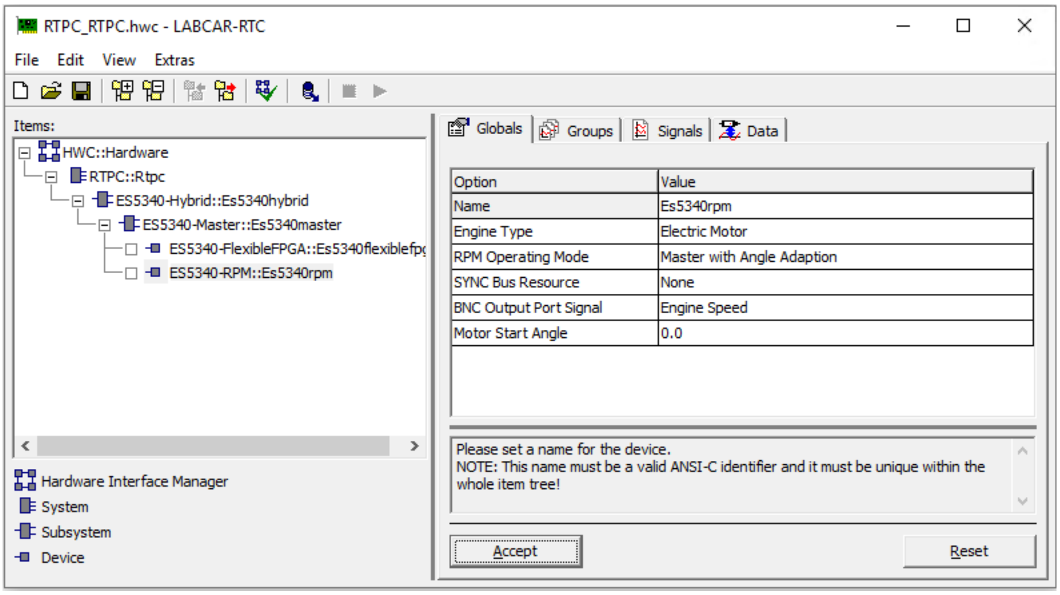
1. Right-click on "ES5340Master" and choose **Add item → ES5340-RPM.**
2. Click **Ok.**



   A new item is added in the hardware editor.

3. Set the parameters in the "Globals" tab according below figure.

   The "Groups" tab contains the tasks for the interfaces to the model. As default, it uses "TaskDVEModel".

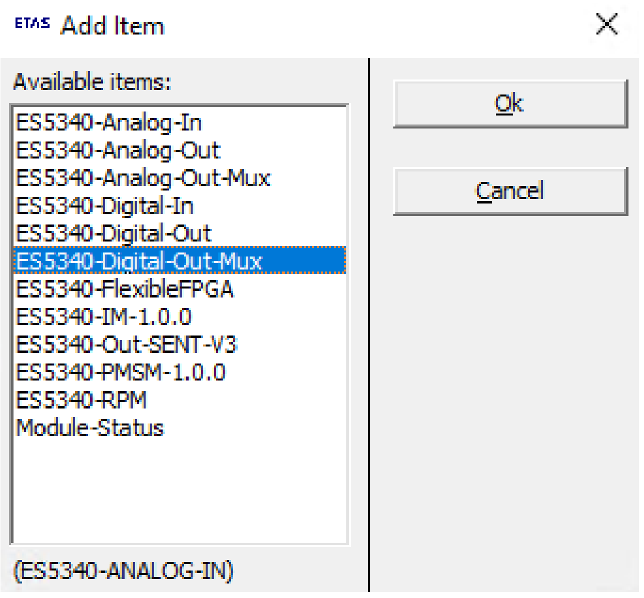

RTPC_RTPC.hwc - LABCAR-RTC
File Edit View Extras
Items:
HWC::Hardware
RTPC::Rtpc
ES5340-Hybrid::Es5340hybrid
ES5340-Master::Es5340master
ES5340-FlexibleFPGA::Es5340flexiblefp
ES5340-RPM::Es5340rpm
Hardware Interface Manager
System
Subsystem
Device
Globals
Groups
Signals
Data
Accept
Reset

Option	Value
Name	Es5340rpm
Engine Type	Electric Motor
RPM Operating Mode	Master with Angle Adaption
SYNC Bus Resource	None
BNC Output Port Signal	Engine Speed
Motor Start Angle	0.0

Please set a name for the device.
NOTE: This name must be a valid ANSI-C identifier and it must be unique within the whole item tree!

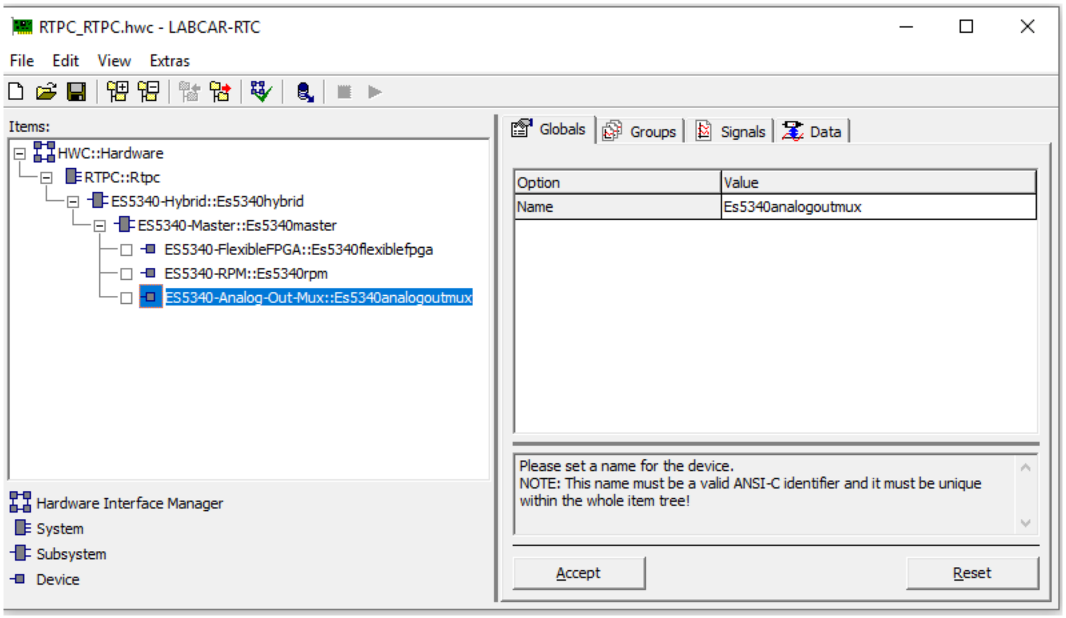
<u>To integrate the Analog-Out-Mux</u>

1. Right-click on "ES5340Master" and choose **Add item → ES5340-Analog-Out-Mux.**
2. Click **Ok.**



A new item is added in the hardware editor.

The "Groups" tab contains the tasks for the interfaces to the model. As default, it uses "TaskDVEModel".



RTPC_RTPC.hwc - LABCAR-RTC
File Edit View Extras
Items:
HWC::Hardware
RTPC::Rtpc
ES5340-Hybrid::Es5340hybrid
ES5340-Master::Es5340master
ES5340-FlexibleFPGA::Es5340flexiblefpga
ES5340-RPM::Es5340rpm
ES5340-Analog-Out-Mux::Es5340analogoutmux
Hardware Interface Manager
System
Subsystem
Device
Globals
Groups
Signals
Data
Accept
Reset

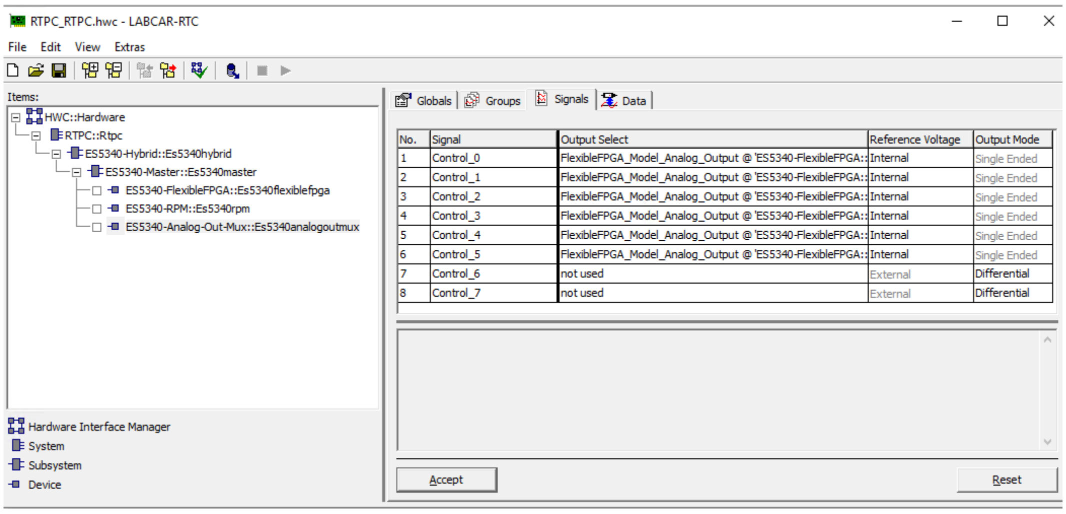
Option	Value
Name	Es5340analogoutmux

Please set a name for the device.
NOTE: This name must be a valid ANSI-C identifier and it must be unique within the whole item tree!

You can set up to 8 channels for the Analog-Out-Mux.

3. For that, configure the according signals from Control_0 to Control_7 in the "Signals" and "Data" tab as shown below.

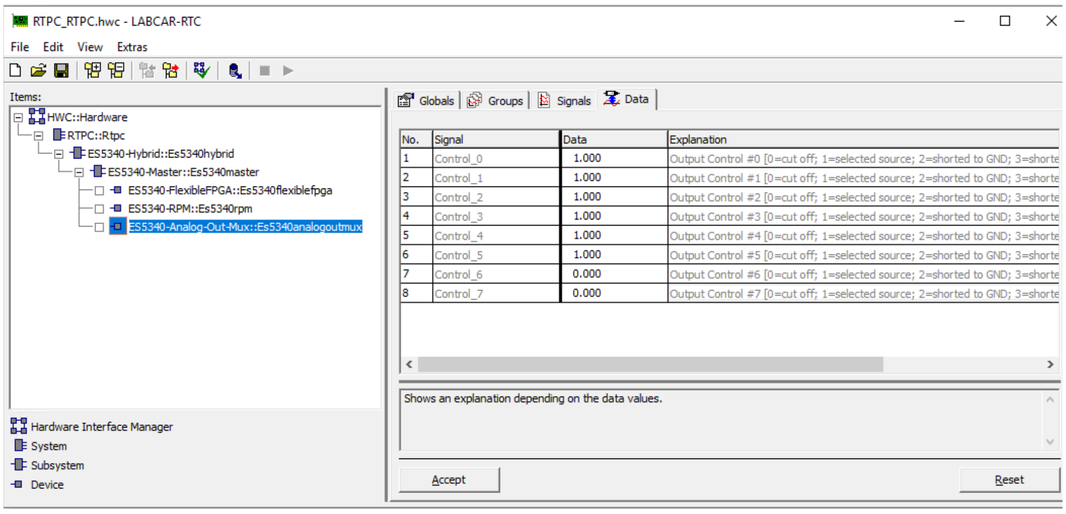
In the example project, only 6 channels are used.



RTPC_RTPC.hwc - LABCAR-RTC
File Edit View Extras
Items:
HWC::Hardware
RTPC::Rtpc
ES5340-Hybrid::Es5340hybrid
ES5340-Master::Es5340master
ES5340-FlexibleFPGA::Es5340flexiblefpga
ES5340-RPM::Es5340rpm
ES5340-Analog-Out-Mux::Es5340analogoutmux
Hardware Interface Manager
System
Subsystem
Device
Globals
Groups
Signals
Data
Accept
Reset

No.	Signal	Output Select	Reference Voltage	Output Mode
1	Control_0	FlexibleFPGA_Model_Analog_Output @ 'ES5340-FlexibleFPGA::	Internal	Single Ended
2	Control_1	FlexibleFPGA_Model_Analog_Output @ 'ES5340-FlexibleFPGA::	Internal	Single Ended
3	Control_2	FlexibleFPGA_Model_Analog_Output @ 'ES5340-FlexibleFPGA::	Internal	Single Ended
4	Control_3	FlexibleFPGA_Model_Analog_Output @ 'ES5340-FlexibleFPGA::	Internal	Single Ended
5	Control_4	FlexibleFPGA_Model_Analog_Output @ 'ES5340-FlexibleFPGA::	Internal	Single Ended
6	Control_5	FlexibleFPGA_Model_Analog_Output @ 'ES5340-FlexibleFPGA::	Internal	Single Ended
7	Control_6	not used	External	Differential
8	Control_7	not used	External	Differential

Fig. 6-1 "Signals" Tab Analog-Out-Mux



RTPC_RTPC.hwc - LABCAR-RTC
File Edit View Extras
Items:
HWC::Hardware
RTPC::Rtpc
ES5340-Hybrid::Es5340hybrid
ES5340-Master::Es5340master
ES5340-FlexibleFPGA::Es5340flexiblefpga
ES5340-RPM::Es5340rpm
ES5340-Analog-Out-Mux::Es5340analogoutmux
Hardware Interface Manager
System
Subsystem
Device
Globals
Groups
Signals
Data
Accept
Reset

No.	Signal	Data	Explanation
1	Control_0	1.000	Output Control #0 [0=cut off; 1=selected source; 2=shorted to GND; 3=shorte
2	Control_1	1.000	Output Control #1 [0=cut off; 1=selected source; 2=shorted to GND; 3=shorte
3	Control_2	1.000	Output Control #2 [0=cut off; 1=selected source; 2=shorted to GND; 3=shorte
4	Control_3	1.000	Output Control #3 [0=cut off; 1=selected source; 2=shorted to GND; 3=shorte
5	Control_4	1.000	Output Control #4 [0=cut off; 1=selected source; 2=shorted to GND; 3=shorte
6	Control_5	1.000	Output Control #5 [0=cut off; 1=selected source; 2=shorted to GND; 3=shorte
7	Control_6	0.000	Output Control #6 [0=cut off; 1=selected source; 2=shorted to GND; 3=shorte
8	Control_7	0.000	Output Control #7 [0=cut off; 1=selected source; 2=shorted to GND; 3=shorte

Shows an explanation depending on the data values.

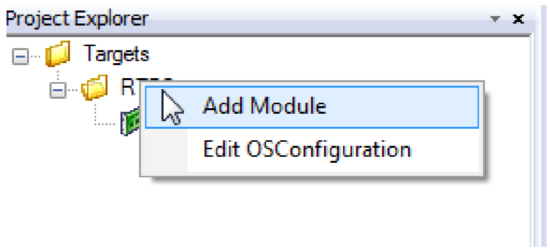
Fig. 6-2 "Data" Tab Analog-Out-Mux

NOTE

Using the Sine encoder and resolver will bring the count of usable Analog Outputs to 6.

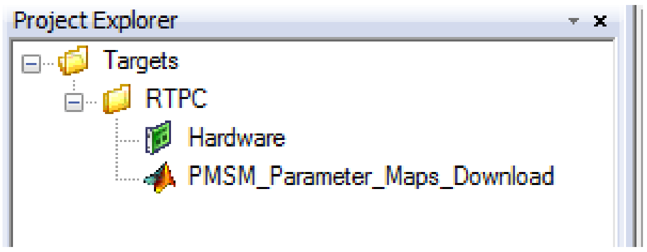
To integrate the Simulink module

1. In the Project Explorer, select "RTPC" under "Targets".
2. Right-click and select **Add Module.**



3. Choose "Add Simulink (TM) Module" and click **Next.**
4. Select "Use existing Simulink (TM) Model (will be copied)" and click **Next**.
5. Use **browse** to `.\Project Items\LCO\PMSM_Param_Download\mdl\Simulink_PMSM_Map_Download\src` in the install folder.
6. Choose "PMSM_Parameter_Maps_Download.mdl" to add the Simulink module.
7. Click **Finish**.

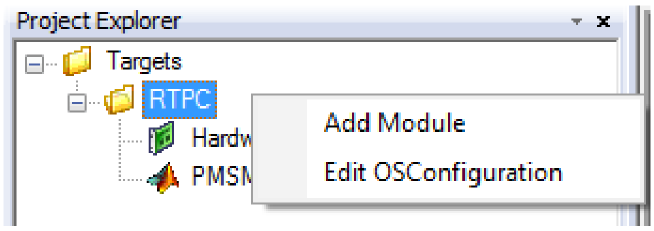
> After successfully adding the Simulink module, the Project Explorer will look like this:



The next step is to integrate a C-code module in order to be able to consider the temperature effect of coils.

To add a C-Code module

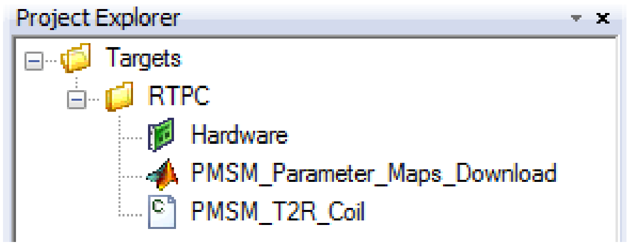
1. In the Project Explorer, select "RTPC" under "Targets".
2. Right-click and select **Add Module.**



3. Choose "Add C-Code Module" and click **Next**.
4. Select "Use existing C-Code Module (will be copied)" and click **Next.**

5. Click **Browse** and go to `.\Project Items\LCO\PMSM_T2R_Coil\cmod\PMSM_T2R_Coil` in the install folder.
6. Choose `PMSM_T2R_Coil.lmd` to add the C-Code module.
7. Click **Finish.**

> After successfully adding the C-Code module, the Project Explorer will look like this.

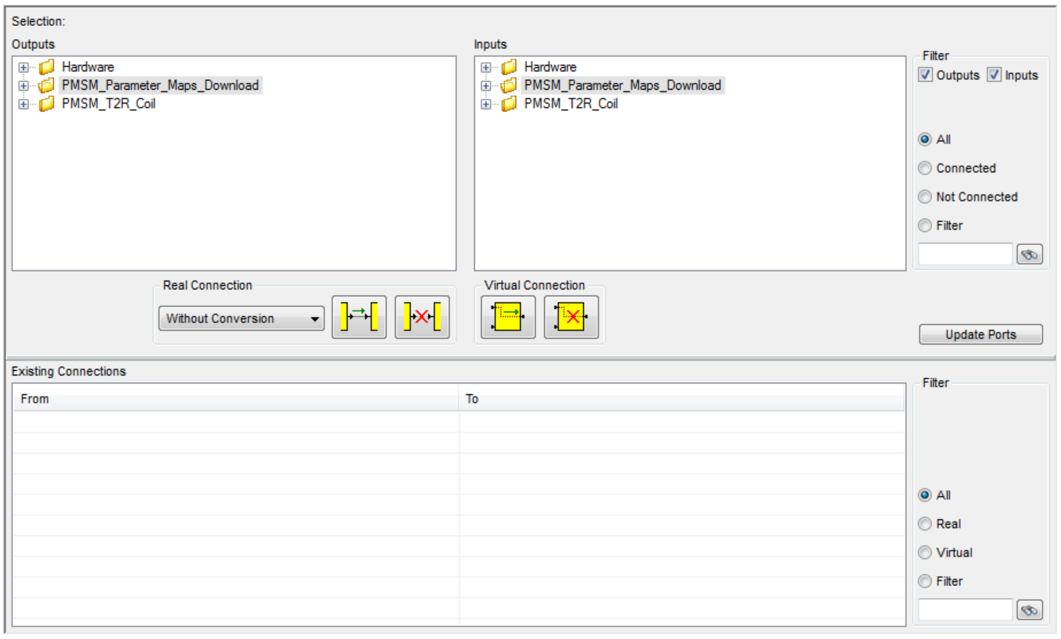


Connection Manager

Per default, the Simulink module and the C-Code module are not connected to the FPGA model. Hence, connection files are used to connect the Simulink module and the C-Code module with the FPGA model.

To connect the modules

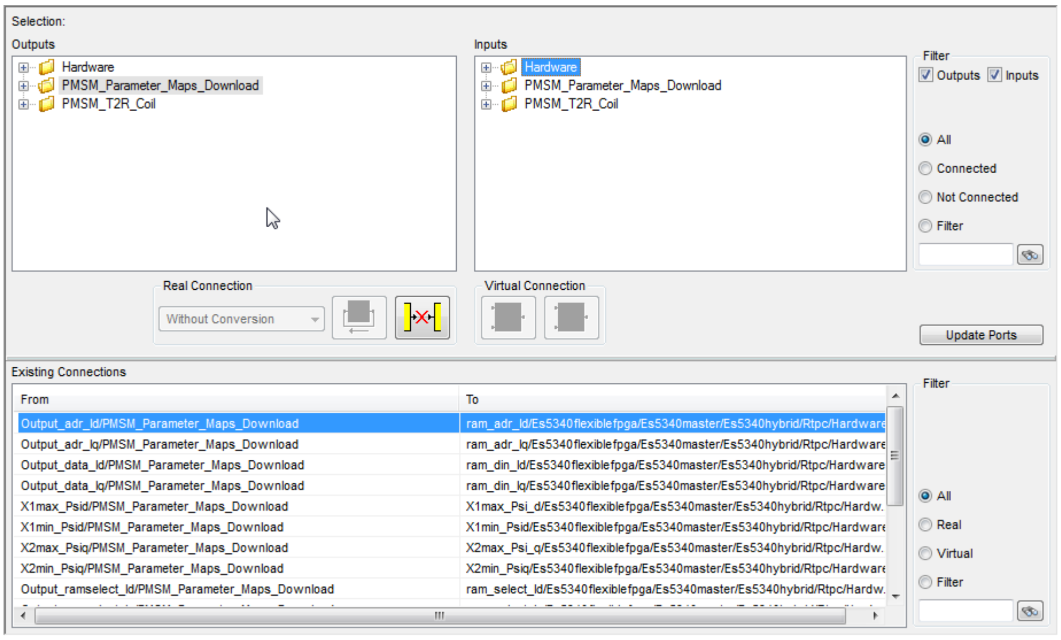
1. Go to the "Connection Manager" tab in the LABCAR-OPERATOR project.



2. Click **Update Ports** in case the interfaces for "Hardware" do not appear in the "Connection Manager" tab.
3. Right click on the "Existing Connections" window and click **Import Connections.**
4. Click **Browse** and go to `.\Project Items\LCO\PMSM_Param_Download\conn` in the install folder.

5. Import the connection file "`PMSM_Parameter_Maps_Download_FPGA.xml`" to establish the connections between PMSM Model and the module.
6. Click **Browse** and go to `.\Project Items\LCO\PMSM_T2R_Coil\conn` in the install folder and import the connection file `PMSM_T2R_Coil_FPGA.xml`.

> After the import of all connections the "Connection Manager" tab will look as follows:



From	To
Output_adr_Id/PMSM_Parameter_Maps_Download	ram_adr_Id/Es5340flexiblefpga/Es5340master/Es5340hybrid/Rtpc/Hardware
Output_adr_Iq/PMSM_Parameter_Maps_Download	ram_adr_Iq/Es5340flexiblefpga/Es5340master/Es5340hybrid/Rtpc/Hardware
Output_data_Id/PMSM_Parameter_Maps_Download	ram_din_Id/Es5340flexiblefpga/Es5340master/Es5340hybrid/Rtpc/Hardware
Output_data_Iq/PMSM_Parameter_Maps_Download	ram_din_Iq/Es5340flexiblefpga/Es5340master/Es5340hybrid/Rtpc/Hardware
X1max_Psid/PMSM_Parameter_Maps_Download	X1max_Psi_d/Es5340flexiblefpga/Es5340master/Es5340hybrid/Rtpc/Hardw.
X1min_Psid/PMSM_Parameter_Maps_Download	X1min_Psid/Es5340flexiblefpga/Es5340master/Es5340hybrid/Rtpc/Hardware
X2max_Psiq/PMSM_Parameter_Maps_Download	X2max_Psi_q/Es5340flexiblefpga/Es5340master/Es5340hybrid/Rtpc/Hardw.
X2min_Psiq/PMSM_Parameter_Maps_Download	X2min_Psiq/Es5340flexiblefpga/Es5340master/Es5340hybrid/Rtpc/Hardware
Output_ramselect_Id/PMSM_Parameter_Maps_Download	ram_select_Id/Es5340flexiblefpga/Es5340master/Es5340hybrid/Rtpc/Hardw.

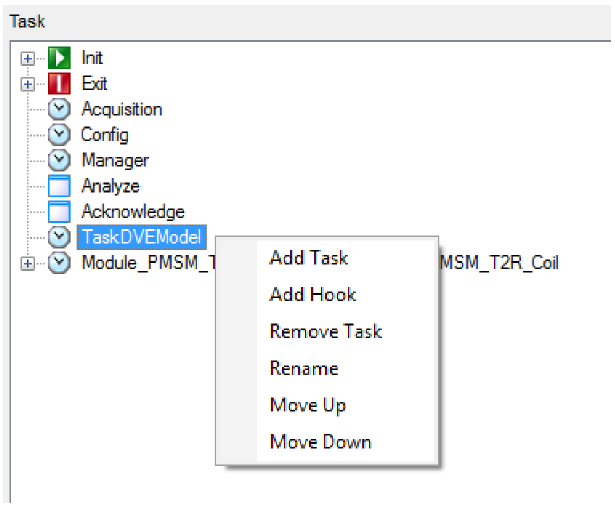
OS Configuration

Using the "OS configuration", you can organize the processes of LABCAR-OPERATOR modules. For the added Simulink module and C-Code module, the order of execution and the selection of a proper step size is important.

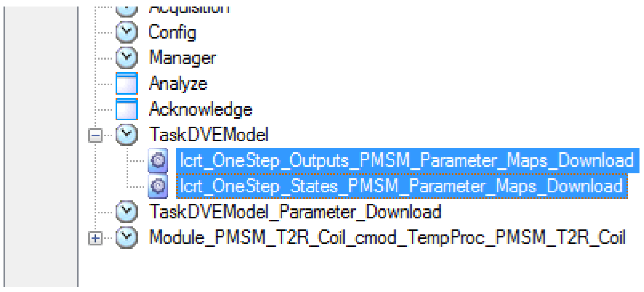
To configure the OS Configuration

1. Go to the "OS Configuration" tab.
2. Click **Update Processes**.
3. Remove unused processes (in gray).

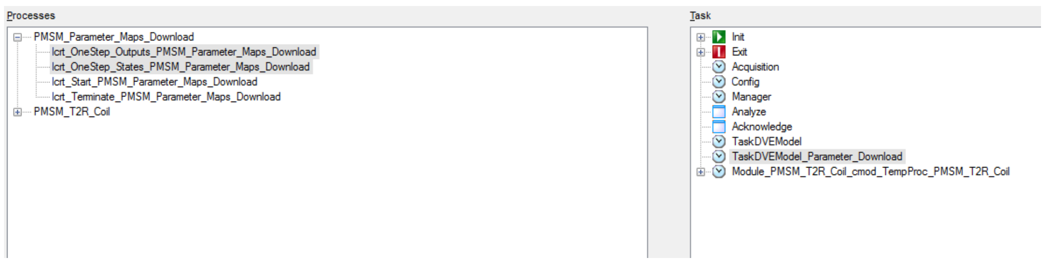
4. Right-click in the task window and choose **Add Task** and rename it to "TaskDVEModel_Parameter_Download".



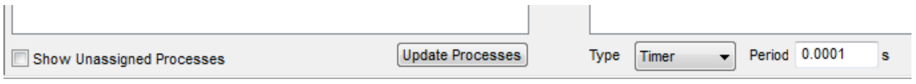
5. Unselect the "Outputs" and "States" of "PMSM_Parameter_-Maps_Download" from the "TaskDVEModel" task and assign to "TaskDVEModel_Parameter_Download".


Acquisition
Config
Manager
Analyze
Acknowledge
TaskDVEModel
lcrt_OneStep_Outputs_PMSM_Parameter_Maps_Download
lcrt_OneStep_States_PMSM_Parameter_Maps_Download
TaskDVEModel_Parameter_Download
Module_PMSM_T2R_Coil_cmod_TempProc_PMSM_T2R_Coil

6. In the "Task" window, select "TaskDVMModel".


Processes
PMSM_Parameter_Maps_Download
lcrt_OneStep_Outputs_PMSM_Parameter_Maps_Download
lcrt_OneStep_States_PMSM_Parameter_Maps_Download
lcrt_Start_PMSM_Parameter_Maps_Download
lcrt_Terminate_PMSM_Parameter_Maps_Download
PMSM_T2R_Coil
Task
Init
Exit
Acquisition
Config
Manager
Analyze
Acknowledge
TaskDVEModel
TaskDVEModel_Parameter_Download
Module_PMSM_T2R_Coil_cmod_TempProc_PMSM_T2R_Coil

7. Set the period to 0.0001.

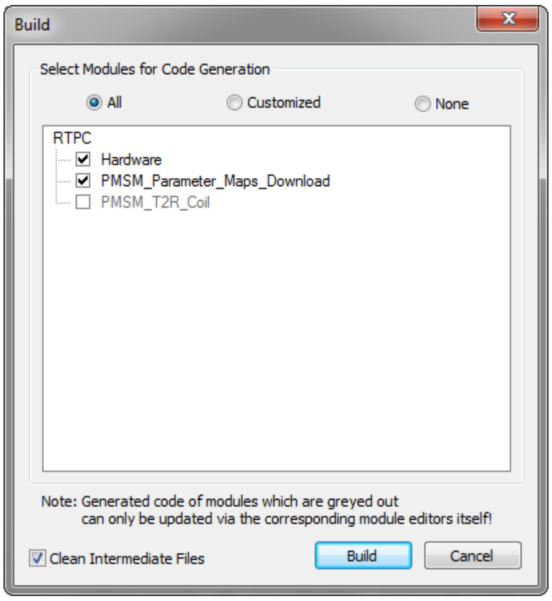


8. In the "Task" window, select "TaskDVEModel_Parameter_Download".
9. Set the period to 0.0004 s.
10. Click **Save All**.

The basic modules to operate the LABACR-MODEL-PMSM basic model are added to the project. In the next step, you can build the LABCAR-OPERATOR project.

To build a LABCAR-OPERATOR project

1. Connect the RTPC.
2. Select **Project** → **Build** with option "All".
3. Click **Build.**



Selected: [✓ All]

RTPC
- [✓] Hardware
- [✓] PMSM_Parameter_Maps_Download
- [ ] PMSM_T2R_Coil

Note: Generated code of modules which are greyed out
can only be updated via the corresponding module editors itself!

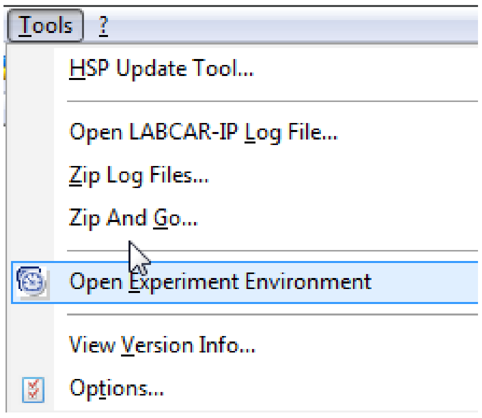
[✓] Clean Intermediate Files

6.2.3 Experiment Creation

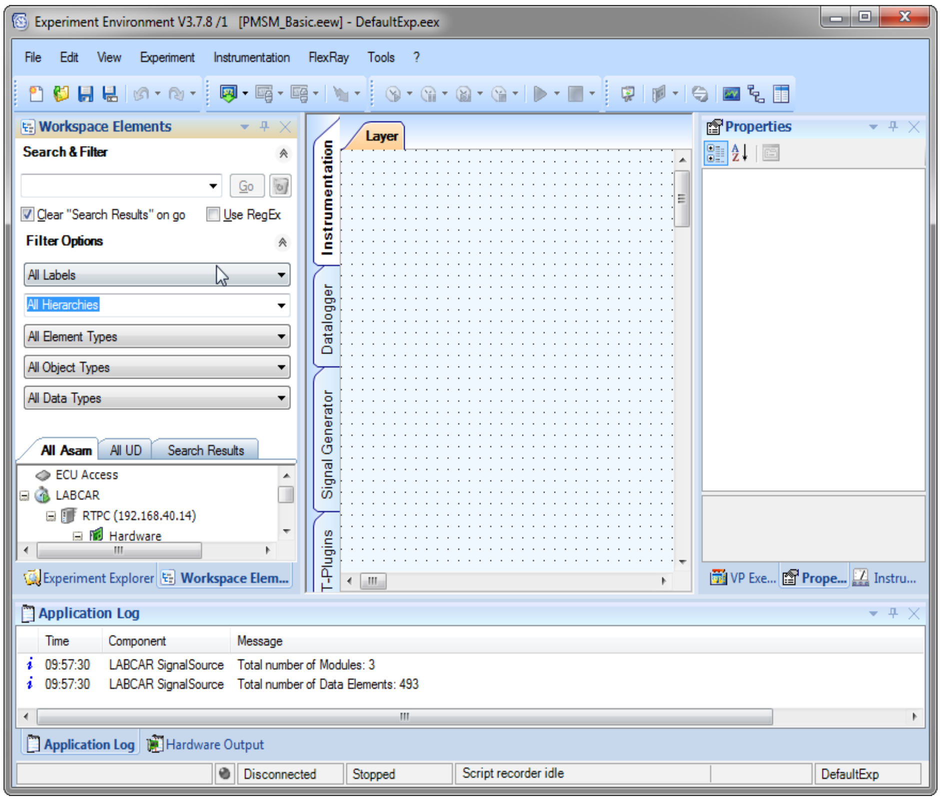
After the successful build of the project, you can prepare the experiment for the first execution of the simulation model. Ensure that the model version flashed as described in section “Flashing Procedure” on page 89 is the same as the model version that is used in the section “Project Creation in LABCAR-OPERATOR” on page 91.

To start LABCAR-EE and to access the LABCAR-MODEL-PMSM ports

1. Open LABCAR-EE using LABCAR-IP
   Tools → **Open Experiment Environment**

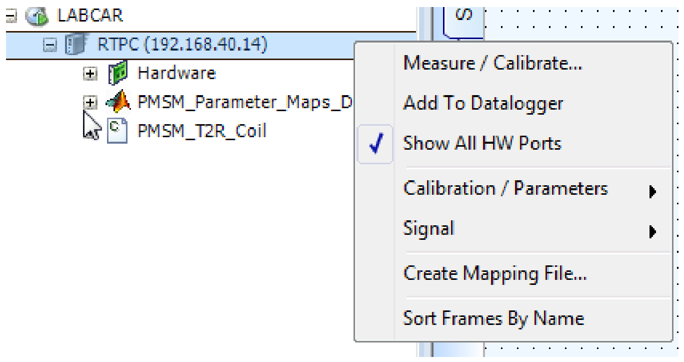


> LACBCAR-EE opens.

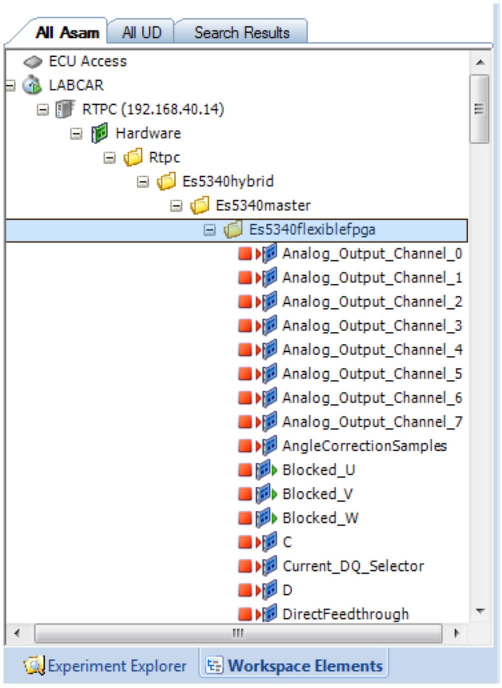


```
Time      Component             Message
09:57:30  LABCAR SignalSource   Total number of Modules: 3
09:57:30  LABCAR SignalSource   Total number of Data Elements: 493
```

2. Select **Experiment → Download → LABCAR.**
3. In “WorkSpaceElements” right-click on **LABCAR → RTPC.**
4. Click **Show All HW Ports.**



> All the ports to the PMSM model are visible.



Parameter Files

No parameter files are provided. The default values of LABCAR-MODEL-PMSM are used.

Layers

The layer provides the basic interface to handle the PMSM model. It allows the setting of the driver mode and the selection of predefined cycles.

NOTE

Please consider that only default LABCAR-EE instruments are used. The layer can be adapted individually to your needs.

The following layers must be installed.

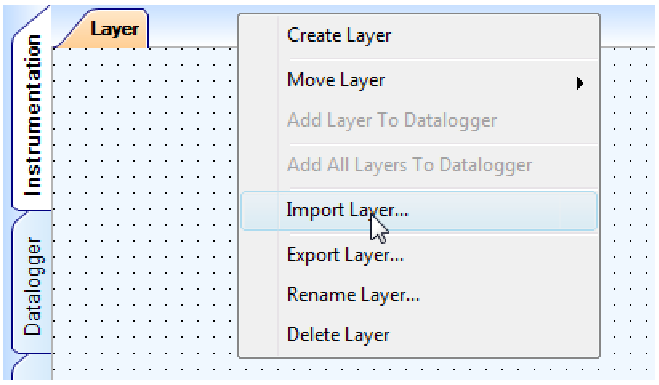
- `PMSM_Simulation_Control.layer`
- `PMSM_Failure_Simulations.layer`
- `PMSM_Parameters_Basic.layer` *or* PMSM_Parameters_Advanced.layer
- PMSM_DAC.layer
- `PMSM_Runtime_License.layer`

You find the layers in the install folder under

- `PMSM_common\gui`
- `02_PMSM_Advanced_specific\gui`
- 03_PMSM_Basic_specific\gui

<u>To add layers</u>

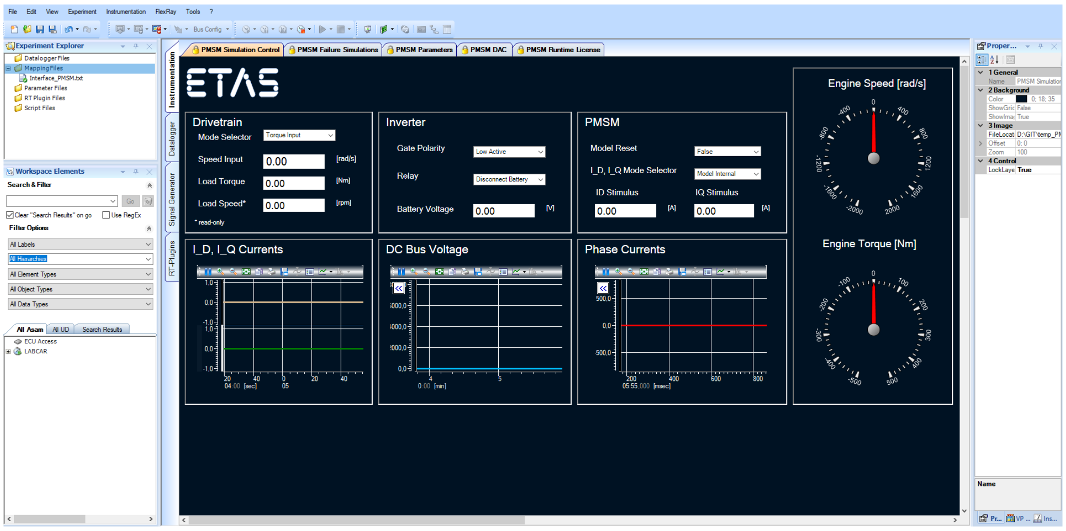
1. Right click on **Instrumentation Layer.**
2. Click **Import Layer** to add layers.



NOTE

All pre-defined instruments use mapped signals provided by the mapping files. For proper execution, ensure that all mapping files are added to the experiment.

> After adding all mapping files, layers, and images, the experimental environment will look like below:



Drivetrain
Mode Selector: Torque Input
Speed Input: 0.00 [rad/s]
Load Torque: 0.00 [Nm]
Load Speed*: 0.00 [rpm]
* read-only

Inverter
Gate Polarity: Low Active
Relay: Disconnect Battery
Battery Voltage: 0.00 [V]

PMSM
Model Reset: False
I_D, I_Q Mode Selector: Model Internal
ID Stimulus: 0.00 [A]
IQ Stimulus: 0.00 [A]

All required artifacts for the first execution of the PMSM_Basic model are added to the experiment. With that, the project can be downloaded to the RTPC. The simulation can be started with the prerequisite of flashing the PMSM model as described in section “Flashing Procedure” on page 89. The section “Getting Started with Tutorial Project” on page 105 describes the handling of the simulation model in more detail.

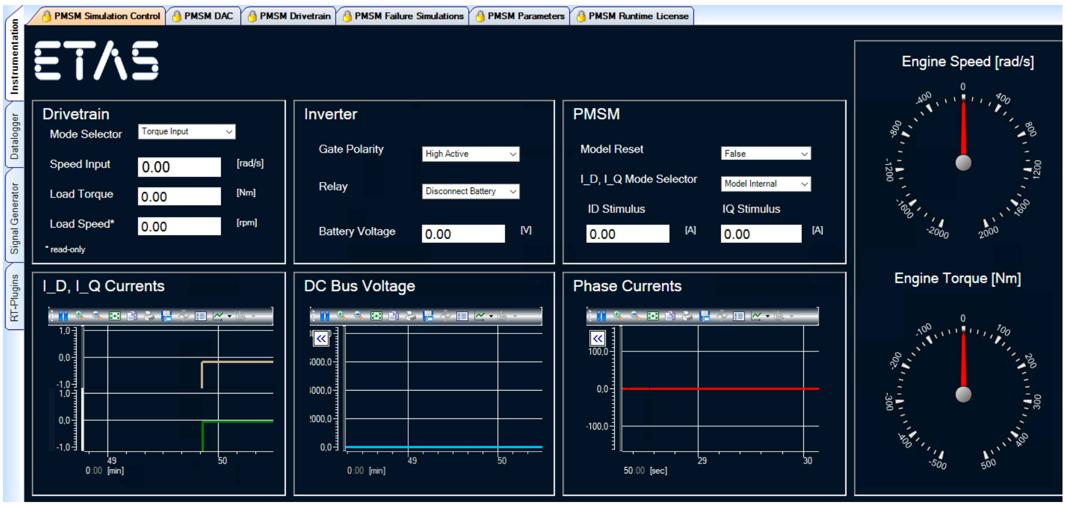
6.3 Getting Started with Tutorial Project

The following tutorial gives an introduction on how to use LABCAR-MODEL-PMSM. The project includes a hardware configuration for a PMSM model running on a ES5340.2-M as well as an example parameter set. To run the model successfully, it is mandatory to flash the ES5340.2-M beforehand with the provided programming files. For further information, see “Flashing Procedure” on page 89.

The tutorial project is located in `.\Example Project\LCO\03_PMSM_Basic` in the install folder.

NOTE

Please do not open or manipulate the example project in the installation folder directly. Instead, copy the example project from the installation folder to a location of your choice.



Drivetrain

Mode Selector	Torque Input	
Speed Input	0.00	[rad/s]
Load Torque	0.00	[Nm]
Load Speed*	0.00	[rpm]

* read-only

Inverter

Gate Polarity	High Active	
Relay	Disconnect Battery	
Battery Voltage	0.00	[V]

PMSM

Model Reset	False	
I_D, I_Q Mode Selector	Model Internal	
ID Stimulus	0.00	[A]
IQ Stimulus	0.00	[A]

6.3.1 Active Short-Circuit Operation Mode

In this example project, LABCAR-MODEL-PMSM is operated under active short condition. This test operating mode does not require any PWM signal actuation to the model. The e-motor is rotated using the speed mode of the drivetrain. The inverter IGBT switches are stimulated using "Fail_IGBT" input of the mode. The following section describes details to run the model in a simple test environment.

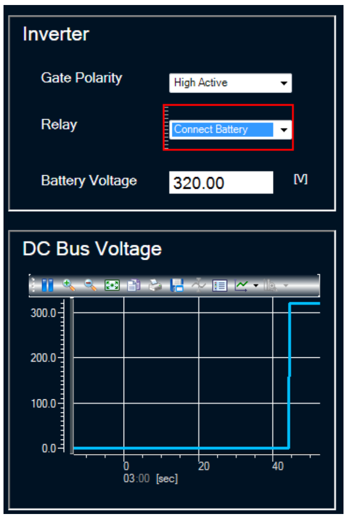
NOTE

Ensure that the PMSM license keys are given in the "PMSM Runtime License" layer and that the status is green or the "Lic_Out" signal is at 1.

1. Connect the relay.

   For example: The battery is connected with the inverter.

   Use the "PMSM Simulation Control" GUI Layer for setting this input.

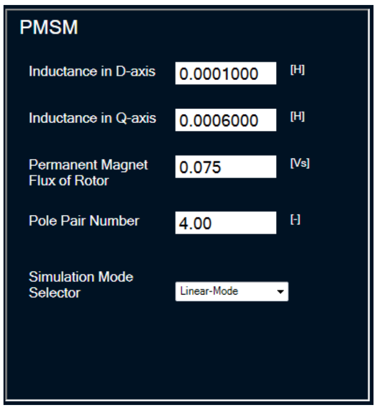
   
   Inverter
   Gate Polarity: High Active
   Relay: Connect Battery
   Battery Voltage: 320.00 [V]
   DC Bus Voltage

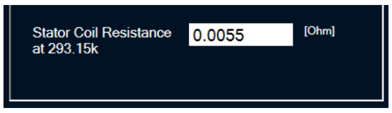
   The intermediate circuit flag can be set to "Disconnected" in "PMSM Parameters" layer. Thus the DC bus voltage is the same as the battery voltage.

2. Use "PMSM Parameters" layer to set the linear mode of the model.

   The input Ld and Lq values are used to compute the Id and Iq values.

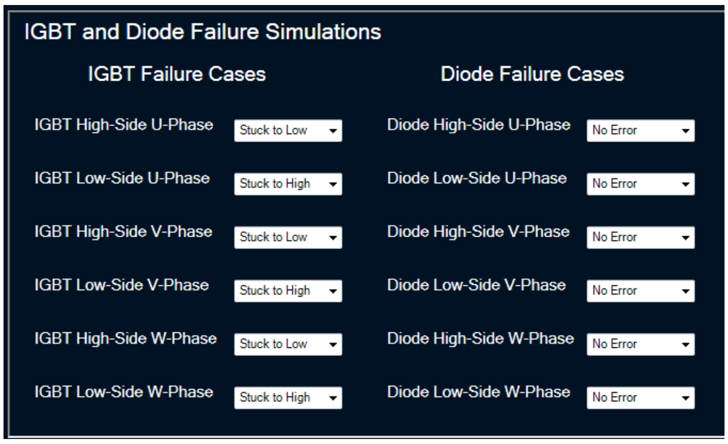
3. Set sample test parameters as shown below.

   
   PMSM
   Inductance in D-axis: 0.0001000 [H]
   Inductance in Q-axis: 0.0006000 [H]
   Permanent Magnet Flux of Rotor: 0.075 [Vs]
   Pole Pair Number: 4.00 [-]
   Simulation Mode Selector: Linear-Mode

   
   Stator Coil Resistance at 293.15k: 0.0055 [Ohm]

4. Use the "PMSM Failure Simulations" layer to set the gate signal input to the switches. Use all the high side switches to stuck to low and the low side switches to stuck to high.

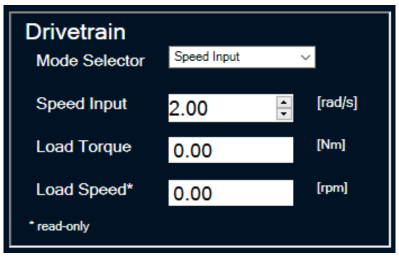
   This essentially simulates an active short circuit.



IGBT and Diode Failure Simulations

IGBT Failure Cases		Diode Failure Cases	
IGBT High-Side U-Phase	Stuck to Low	Diode High-Side U-Phase	No Error
IGBT Low-Side U-Phase	Stuck to High	Diode Low-Side U-Phase	No Error
IGBT High-Side V-Phase	Stuck to Low	Diode High-Side V-Phase	No Error
IGBT Low-Side V-Phase	Stuck to High	Diode Low-Side V-Phase	No Error
IGBT High-Side W-Phase	Stuck to Low	Diode High-Side W-Phase	No Error
IGBT Low-Side W-Phase	Stuck to High	Diode Low-Side W-Phase	No Error

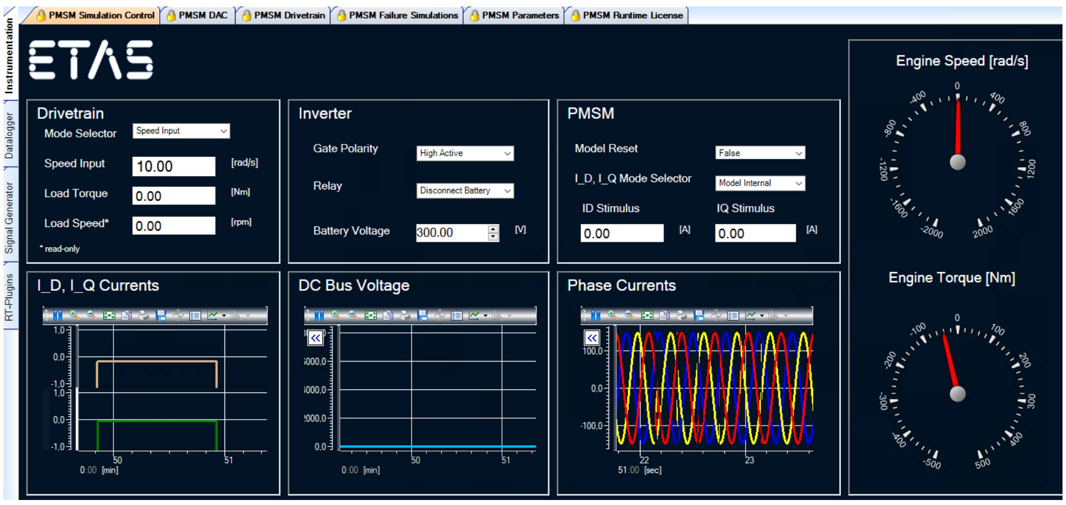
5. Use the Speed Input of the Drivetrain to achieve a positive speed.



Drivetrain

Mode Selector	Speed Input	
Speed Input	2.00	[rad/s]
Load Torque	0.00	[Nm]
Load Speed*	0.00	[rpm]

* read-only



Drivetrain

Mode Selector	Speed Input	
Speed Input	10.00	[rad/s]
Load Torque	0.00	[Nm]
Load Speed*	0.00	[rpm]

* read-only

Inverter

Gate Polarity	High Active	
Relay	Disconnect Battery	
Battery Voltage	300.00	[V]

PMSM

Model Reset	False	
I_D, I_Q Mode Selector	Model Internal	
ID Stimulus	0.00	[A]
IQ Stimulus	0.00	[A]

In the image above you see that the currents flow in the motor and the breaking torque is generated i.e., a negative torque is generated.

6.4 Nonlinear Maps and Parameter Maps Downloader

The saturation of the machine core is considered using maps of inductance as a function of currents i.e., *Ld* and *Lq* as *f (Id , Iq)*. LABCAR-MODEL uses an entire-flux model for simulating saturation effects, so that an inversion of the inductance maps is required. Hence, the current maps as functions of fluxes I_d (φ_d, φ_q) and I_q (φ_d, φ_q) are required.

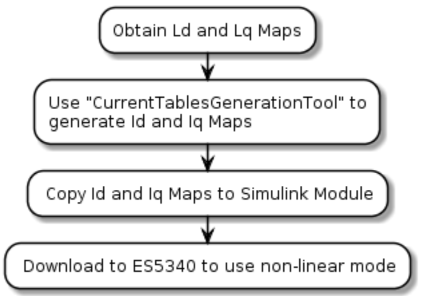
Here

L_d and L_q are inductance [H] along d and q axis respectively.

I_d and I_q are currents [A] along d and q axis respectively.

φ_d and φ_q are fluxes [Vs] along d and q axis respectively.

The section consists of two parts. The first part is a MATLAB based tool to invert the inductance data maps to obtain the current data maps. The second part is a Simulink module to download the current data maps to the ES5340.2-M.


Obtain Ld and Lq Maps
Use "CurrentTablesGenerationTool" to generate Id and Iq Maps
Copy Id and Iq Maps to Simulink Module
Download to ES5340 to use non-linear mode

A MATLAB based tool "`CurrentTablesGenerationTool.m`" is provided to invert the inductance tables (*f(Id , Iq)*) to obtain lookup tables for I_d and I_q (f(φ_d and φ_q)). The tool "`CurrentTablesGenerationTool.m`" is located at `.\FPGA Model\PMSM_Param_Download\EMachineData_PreProcessing`.

You can provide the inductance lookup table data (shall not contain any complex numbers) L_d and L_q with row break points of I_d and column break points of I_q to a dedicated file "`EngineInputData.m`". The lookup tables are written to "`GeneratedTables.txt`". The data can be further used with a Simulink module to download to FPGA.

The tool has the following important characteristics.

- Equidistant and identical breakpoints in d-direction (φ_d) and q-direction (φ_q) of lookup tables.
- The distance of breakpoints is a power of two for powerful FPGA usage.
- No extrapolation i.e. means that at the end point the map is clipped.
- Bilinear interpolation with possibility of different interpolation methods.
- The lookup tables data is provided for positive and negative axis for (φ_d) for simulation performance reasons.

- The maximum size of each lookup table is the number of rows multiplied with the number of currents. It is more or less equal to 52200 (Maximum number of data points allowed on the FPGA RAM).

A Simulink module "PMSM_Parameter_Maps_Download" is provided to download the current maps to the ES5340.2-M. The connection file "`PMSM_Parameter_Maps_Download_FPGA.xml`" consists of the connection between the Simulink module and the FPGA model. The Simulink module needs to be added to the project and the connections need to be imported. The section “Project and Experiment Creation” on page 88 describes how to add the Simulink module and how to import the connections.

The Simulink module consists of a file "`PMSM_Parameter_Maps.m`" to provide current maps of I_d and I_q and the break points vectors φ_d and φ_q.

The current maps and break points needs to be obtained from the "`CurrentTablesGenerationTool.m`" discussed above. "`GeneratedTables.txt`" consists of the current maps data and break points data. This data needs to be entered to "`PMSM_Parameter_Maps.m`". By setting the Simulink module input "`RAM_Write_Trigger`" to 1, the writing of data fields to the FPGA is enabled. When writing the data fields to the ES5340.2-M board, the FPGA model input "`Reset`" is set to 1.

Use the "`Linear_NonLinear_Selector`" input of the FPGA model to use the non-linear mode of the simulation.

6.5 Signal Interface

In the following, the basic interfaces to operate LABCAR-MODEL-PMSM are given.

6.5.1 Drivetrain Interfaces

Inputs

Name	Type	Unit	Description
T_L	Fixdt(1,32,21)	Nm	Load torque
Phi_mech_Init	Fixdt(0,32,28)	rad	Initial mechanical rotor angle
Omega_Input	Fixdt(1,32,15)	rad/s	Speed input of drivetrain in speed mode

Outputs

Name	Type	Unit	Description	DAC Multiplexer Input Number
Omega_mech	Fixdt(1,32,15)	rad/s	Mechanical rotor speed	1
Omega_el	Fixdt(1,32,15)	rad/s	Electrical rotor speed	0
Phi_mech_rad	Fixdt(0,18,15)	rad	Mechanical rotor angle	3
Phi_el	Fixdt(1,18,15)	rad	Electrical rotor angle	2
T_mech	Fixdt(1,32,21)	Nm	Resulting mechanical rotor torque	4
Tel_average	Fixdt(1,32,18)	Nm	Average electrical torque	5
n_load_rpm	Int32	rpm	Mechanical load speed	27
n_rotor_rpm	Int32	rpm	Mechanical rotor speed	-
phi_rotor_deg	Uint16	degree	Mechanical rotor angle	-

Parameters

Name	Type	Unit	Description	Default	Range
f	Fixdt(0,32,28)	Nms/ rad	Friction coefficient of E-machine; to deactivate set parameter to zero value	0.1	0.0...10
C	Fixdt(0,32,8)	Nm/ rad	Torsional stiffness between rotor and load; to deactivate set parameter to zero value	1500000	0.0...100 00000
D	Fixdt(0,32,22)	Nms/ rad	Damping between rotor and load; to deactivate set parameter to zero value	100.0	0.0...100
Direct Feed through	Boolean	-	mode selector 0: use torque input 1: use speed input	0	[0,1]
invJ_E	Fixdt(0,32,12)	1/ kgm2	Inverse of inertia of rotor	10	0.1...1e5
invJ_L	Fixdt(0,32,12)	1/ kgm2	Inverse of inertia of load/drive-train	10	0.1...1e5

6.5.2 Inverter Interfaces

Inputs

Name	Type	Unit	Description
PWM_UH	Boolean	-	Control signal upper IGBT phase U
PWM_UL	Boolean	-	Control signal lower IGBT phase U
PWM_VH	Boolean	-	Control signal upper IGBT phase V
PWM_VL	Boolean	-	Control signal lower IGBT phase V
PWM_WH	Boolean	-	Control signal upper IGBT phase W
PWM_WL	Boolean	-	Control signal lower IGBT phase W
Gate_Signal_Polarity	Boolean	-	IGBT Polarity Control 0: Low active, conducting while gate is "Low" 1: High active, conducting while gate is "High"
Fail_IGBT_HS_U	Fixdt(0,2,0)	-	Gate failure mode of high side switch of phase U 0: Normal Operation 1: Stuck on "Low" 2: Stuck on "High"
Fail_IGBT_HL_U	Fixdt(0,2,0)	-	Gate failure mode of low side switch of phase U 0: Normal Operation 1: Stuck on "Low" 2: Stuck on "High"
Fail_IGBT_HS_V	Fixdt(0,2,0)	-	Gate failure mode of low side switch of phase V 0: Normal Operation 1: Stuck on "Low" 2: Stuck on "High"

Name	Type	Unit	Description
Fail_IGBT_HL_V	Fixdt(0,2,0)	-	Gate failure mode of high side switch of phase V 0: Normal Operation 1: Stuck on "Low" 2: Stuck on "High"
Fail_IGBT_HS_W	Fixdt(0,2,0)	-	Gate failure mode of high side switch of phase W 0: Normal Operation 1: Stuck on "Low" 2: Stuck on "High"
Fail_IGBT_HL_W	Fixdt(0,2,0)	-	Gate failure mode of low side switch of phase W 0: Normal Operation 1: Stuck on "Low" 2: Stuck on "High"
Fail_Diode_HS_U	Fixdt(0,2,0)	-	Diode failure mode of high side switch of phase U 0: Normal Operation 1: Stuck on "Low" 2: Stuck on "High"
Fail_Diode_HL_U	Fixdt(0,2,0)	-	Diode failure mode of low side switch of phase U 0: Normal Operation 1: Stuck on "Low" 2: Stuck on "High"
Fail_Diode_HS_V	Fixdt(0,2,0)	-	Diode failure mode of high side switch of phase V 0: Normal Operation 1: Stuck on "Low" 2: Stuck on "High"
Fail_Diode_HL_V	Fixdt(0,2,0)	-	Diode failure mode of low side switch of phase V 0: Normal Operation 1: Stuck on "Low" 2: Stuck on "High"
Fail_Diode_HS_W	Fixdt(0,2,0)	-	Diode failure mode of high side switch of phase W 0: Normal Operation 1: Stuck on "Low" 2: Stuck on "High"
Fail_Diode_HL_W	Fixdt(0,2,0)	-	Diode failure mode of low side switch of phase W 0: Normal Operation 1: Stuck on "Low" 2: Stuck on "High"
U_Batt0	Fixdt(0,16,5)	V	Input voltage of battery

Name	Type	Unit	Description
RC_Ckt_Disable_Flag	Boolean	-	Intermediate circuit disable flag 0: Intermediate circuit enabled 1: Intermediate circuit disabled
Relay_Switch	Boolean	-	Switch to connect/disconnect battery 0: Battery disconnected 1: Battery connected
ShortCircuit_Flag_Reset	Boolean	-	Input for resetting short circuit error flags

Outputs

Name	Type	Unit	Description	DAC Multiplexer Input Number
U_DC	Fixdt(1,32,18)	V	Intermediate circuit voltage	19
I_Bat	Fixdt(1,32,18)	A	Battery current	-
I_DC	Fixdt(1,18,4)	A	Intermediate circuit current	20
U_phase_U	Fixdt(1,32,18)	V	Voltage phase U	9
U_phase_V	Fixdt(1,32,18)	V	Voltage phase V	10
U_phase_W	Fixdt(1,32,18)	V	Voltage phase W	11
ShortCircuit_Error_-Flag_U	Boolean	-	Short circuit error flag for half bridge phase U	-
ShortCircuit_Error_-Flag_V	Boolean	-	Short circuit error flag for half bridge phase V	-
ShortCircuit_Error_-Flag_W	Boolean	-	Short circuit error flag for half bridge Phase W	-
I_element_H_U	Fixdt(1,16,4)	A	Current through upper switch phase U	-
I_element_H_V	Fixdt(1,16,4)	A	Current through upper switch phase V	-
I_element_H_W	Fixdt(1,16,4)	A	Current through upper switch phase W	-
I_element_L_U	Fixdt(1,16,4)	A	Current through lower switch phase U	-
I_element_L_V	Fixdt(1,16,4)	A	Current through lower switch phase V	-
I_element_L_W	Fixdt(1,16,4)	A	Current through lower switch phase W	-
U_Switch_H_U	Fixdt(1,32,18)	V	Voltage on upper switch phase U	-
U_Switch_H_V	Fixdt(1,32,18)	V	Voltage on upper switch phase V	-
U_Switch_H_W	Fixdt(1,32,18)	V	Voltage on upper switch phase W	-
U_Switch_L_U	Fixdt(1,32,18)	V	Voltage on lower switch phase U	-
U_Switch_L_V	Fixdt(1,32,18)	V	Voltage on lower switch phase V	-
U_Switch_L_W	Fixdt(1,32,18)	V	Voltage on lower switch phase W	-
Blocked_U	Boolean	-	Flag set if both upper and lower switches of phase U are open	-
Blocked_V	Boolean	-	Flag set if both upper and lower switches of phase V are open	-
Blocked_W	Boolean	-	Flag set if both upper and lower switches of phase W are open	-

Parameters

Name	Type	Unit	Description	Default	Range
U_D_Fwd	Fixdt(0,16,14)	V	Conducting voltage in diode (293.15K)	0.2	0.01...3
Inv_R_Batt	Fixdt(0,16,12)	1/Ω	Inverse of inner resistance of battery	0.5	1e-3...10
Inv_C_DC	Fixdt(0,16,12)	1/µF	Inverse of intermediate circuit capacitor	0.1	1e-3...0.1
T_Switch [1]	Fixdt(0,16,32)	s	Switching time of IGBT	5e-6	0.2e-6 ...10e -6
R_SW_ON [1]	Fixdt(0,16,20)	Ω	Conducting resistance in IGBT	0.001	0.0...0.05
R_D_FWD [1]	Fixdt(0,16,20)	Ω	Conducting resistance in diode	0.001	0.0...0.05
U_SW_Fwd [1]	Fixdt(0,16,14)	V	Conducting voltage in IGBT(293.15K)	0.2	0.01...3

[1] This signal is only available in LABCAR-MODEL-PMSM Advanced.

NOTE

The time constant of the intermediate circuit (`R_Batt * C_DC`) must be greater than 1µs.

6.5.3 PMSM Motor Interfaces

Inputs

Name	Type	Unit	Description
Rs	Single	Ω	Stator coil resistance
Riron [2]	Single	Ω	Resistance represents iron losses of machine
L_d	Single	H	Inductance in d-axis
L_q	Single	H	Inductance in q-axis
Psi_Rotor	Single	Wb (Vs)	Permanent magnetic flux in rotor
Linear_NonLinear_Selector	Boolean	-	0: linear mode, inputs "L_d" and "L_q" are used. 1: Nonlinear mode, nonlinear maps Id(Psi_d, Psi_q) and Iq(Psi_d, Psi _q) are used.
Current_DQ_Selector	Boolean	-	0: Internally calculated values for I_d and I_q are used. 1: Stimulus inputs "ID_Stimu-late" and "IQ_Stimulate" are used.
ID_Stimulate	Single	A	Stimulating input for I_d
IQ_Stimulate	Single	A	Stimulating input for I_q
Reset	Boolean	-	Reset all internal states
AngleCorrectionSamples	Fixdt(0,8,2)	-	Number of samples consid-ered for angle correction of I_d and I_q calculations
RAM_Write_Prog	Boolean	-	Indication if data is written to FPGA RAM
ram_select_Id	Uint16	-	Selector to write data into the d-axis RAM block
ram_adr_Id	Uint32	-	RAM address to write in d-axis RAM block
ram_din_Id	Int32	-	Data for d-axis
ram_select_Iq	Uint16	-	Selector to write data into the q-axis RAM block
ram_adr_Iq	Uint32	-	RAM address to write in q-axis RAM block
ram_din_Iq	Int32	-	Data for q-axis
X1min_Psid	Fixdt(1,36,26)	Wb (Vs)	Minimum break point in d-axis direction
X2min_Psiq	Fixdt(1,36,26)	Wb (Vs)	Minimum break point in q-axis direction
discr1_Psid	Int8	-	Distance between break points in d-axis direction
discr2_Psiq	Int8	-	Distance between break points in q-axis direction

Name	Type	Unit	Description
N1_Psi_d	Uint16	-	Number of break points in d-axis direction
N2_Psi_q	Uint16	-	Number of break points in q-axis direction
X1max_Psi_d	Fixdt(1,36,26)	Wb (Vs)	Maximum break point in d-axis direction
X2max_Psi_q	Fixdt(1,36,26)	Wb (Vs)	Maximum break point in q-axis direction

[2] This signal is only available in LABCAR-MODEL-PMSM Advanced.

Outputs

Name	Type	Unit	Description	DAC Multiplexer Input Number
Vd	Single	V	Voltage in d-axis	-
Vq	Single	V	Voltage in q-axis	-
Psi_d	Single	Wb (Vs)	Flux in d-axis	17
Psi_q	Single	Wb (Vs)	Flux in q-axis	18
Torque	Single	Nm	Motor electromagnetic torque	-
U_Ind_U	Single	V	Induced back EMF of phase U	14
U_ind_V	Single	V	Induced back EMF of phase V	15
U_ind_W	Single	V	Induced back EMF of phase W	16
I_Phase_U	Single	A	Current of phase U	6
I_Phase_V	Single	A	Current of phase V	7
I_Phase_W	Single	A	Current of phase W	8
Id	Single	A	Current in d-axis	12
Iq	Single	A	Current in q-axis	13
PLoss_Copper	Single	W	Copper loss of motor	25
PLoss_Core	Single	W	Core loss of motor	26
Icd	Single	A	Current in core loss branch of d-axis	21
Icq	Single	A	Current in core loss branch q-axis	22
Imd	Single	A	Current in magnetization branch of d-axis	23
Imq	Single	A	Current in magnetization branch of q-axis	24

Parameters

Name	Type	Unit	Description	Default	Range
PPN	Uint8	-	Number of pole pairs	1	1...32

6.5.4 License Keys Interface

Inputs

Name	Type	Unit	Description
Lic_0	Uint32	-	License key number 0
Lic_1	Uint32	-	License key number 1
Lic_2	Uint32	-	License key number 2
Lic_3	Uint32	-	License key number 3
Lic_4	Uint32	-	License key number 4

Output

Name	Type	Unit	Description	DAC Multiplexer Input Number
Lic_Out	Boolean	-	Indication if license keys are correct	-

6.5.5 Analog Outputs

The LABCAR-MODEL-PMSM supports the 8 analog outputs of the ES5340.2-M board.

NOTE

The resolver sensor, which is also implemented in the FPGA on the ES5340.2-M, uses the analog outputs #6 and #7. When it is intended to use a resolver sensor, the analog outputs #6 and #7 are unavailable to the FPGA model.

Each electrical output channel of the ES5340.2-M can be connected to any other signal of the simulation model by using the multiplexer as described in the column "DAC Multiplexer Input Number". By choosing the multiplexer input number, a particular model signal can be connected to the analog output channel. Each output channel can be configured further on by parameters, which can influence the output voltage of the board that are represented in the table below.

Additionally the following demonstration provides an overview about the interfaces and the selectable signals of LABCAR-MODEL-PMSM.

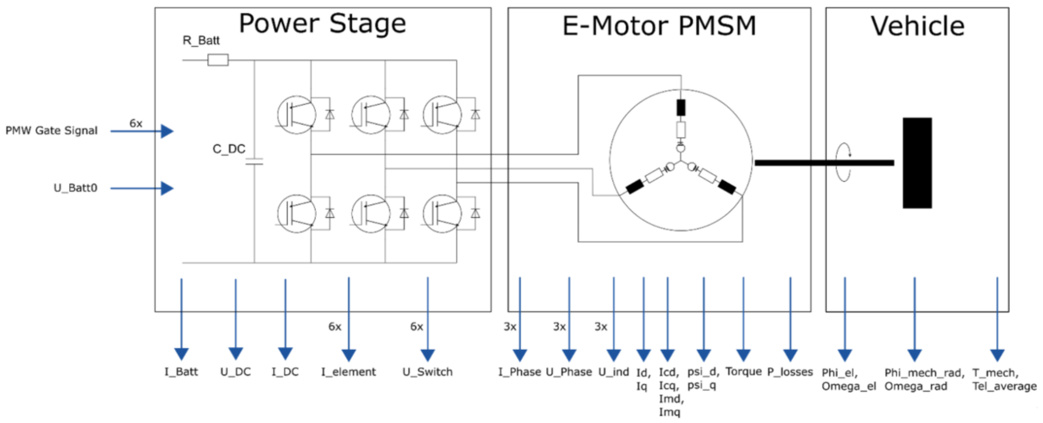


Fig. 6-3 LABCAR-MODEL-PMSM Model Interface Overview

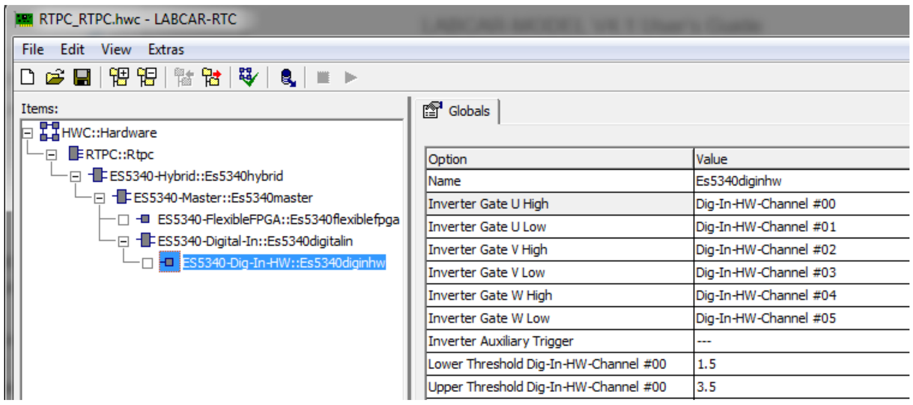
Name	Type	Unit	Description
ScalingFactor_AnaOut_0	Fixdt(1,20,17)	-	Scaling factor for scaling output to 10 V of analog channel 0
ScalingFactor_AnaOut_1	Fixdt(1,20,17)	-	Scaling factor for scaling output to 10 V of analog channel 1
ScalingFactor_AnaOut_2	Fixdt(1,20,17)	-	Scaling factor for scaling output to 10 V of analog channel 2
ScalingFactor_AnaOut_3	Fixdt(1,20,17)	-	Scaling factor for scaling output to 10 V of analog channel 3
ScalingFactor_AnaOut_4	Fixdt(1,20,17)	-	Scaling factor for scaling output to 10 V of analog channel 4
ScalingFactor_AnaOut_5	Fixdt(1,20,17)	-	Scaling factor for scaling output to 10 V of analog channel 5
ScalingFactor_AnaOut_6	Fixdt(1,20,17)	-	Scaling factor for scaling output to 10 V of analog channel 6
ScalingFactor_AnaOut_7	Fixdt(1,20,17)	-	Scaling factor for scaling output to 10 V of analog channel 7
Offset_AnaOut_0	Fixdt(1,16,11)	V	Offset for setting voltage level of analog output channel 0
Offset_AnaOut_1	Fixdt(1,16,11)	V	Offset for setting voltage level of analog output channel 1
Offset_AnaOut_2	Fixdt(1,16,11)	V	Offset for setting voltage level of analog output channel 2
Offset_AnaOut_3	Fixdt(1,16,11)	V	Offset for setting voltage level of analog output channel 3
Offset_AnaOut_4	Fixdt(1,16,11)	V	Offset for setting voltage level of analog output channel 4
Offset_AnaOut_5	Fixdt(1,16,11)	V	Offset for setting voltage level of analog output channel 5
Offset_AnaOut_6	Fixdt(1,16,11)	V	Offset for setting voltage level of analog output channel 6
Offset_AnaOut_7	Fixdt(1,16,11)	V	Offset for setting voltage level of analog output channel 7

Name	Type	Unit	Description
Max_AnaOut_0	Fixdt(1,16,11)	V	For setting maximum output voltage level of board of channel 0
Max_AnaOut_1	Fixdt(1,16,11)	V	For setting maximum output voltage level of board of channel 1
Max_AnaOut_2	Fixdt(1,16,11)	V	For setting maximum output voltage level of board of channel 2
Max_AnaOut_3	Fixdt(1,16,11)	V	For setting maximum output voltage level of board of channel 3
Max_AnaOut_4	Fixdt(1,16,11)	V	For setting maximum output voltage level of board of channel 4
Max_AnaOut_5	Fixdt(1,16,11)	V	For setting maximum output voltage level of board of channel 5
Max_AnaOut_6	Fixdt(1,16,11)	V	For setting maximum output voltage level of board of channel 6
Max_AnaOut_7	Fixdt(1,16,11)	V	For setting maximum output voltage level of board of channel 7
Min_AnaOut_0	Fixdt(1,16,11)	V	For setting minimum output voltage level of board of channel 0
Min_AnaOut_1	Fixdt(1,16,11)	V	For setting minimum output voltage level of board of channel 1
Min_AnaOut_2	Fixdt(1,16,11)	V	For setting minimum output voltage level of board of channel 2
Min_AnaOut_3	Fixdt(1,16,11)	V	For setting minimum output voltage level of board of channel 3
Min_AnaOut_4	Fixdt(1,16,11)	V	For setting minimum output voltage level of board of channel 4
Min_AnaOut_5	Fixdt(1,16,11)	V	For setting minimum output voltage level of board of channel 5
Min_AnaOut_6	Fixdt(1,16,11)	V	For setting minimum output voltage level of board of channel 6
Min_AnaOut_7	Fixdt(1,16,11)	V	For setting minimum output voltage level of board of channel 7

6.5.6 Digital Inputs

LABCAR-MODEL-PMSM requires 6 digital PWM input signals for controlling the IGBTs of the inverter. These signals need to be added to the project using in the Hardware Configuration. Please see the LABCAR-OPERATOR manual for details [1].

The "Digital Input of hardware" channel can be connected to LABCAR-MODEL-PMSM using "hwc" as shown below.



RTPC_RTPC.hwc - LABCAR-RTC

Items:

- HWC::Hardware
  - RTPC::Rtpc
    - ES5340-Hybrid::Es5340hybrid
      - ES5340-Master::Es5340master
        - ES5340-FlexibleFPGA::Es5340flexiblefpga
        - ES5340-Digital-In::Es5340digitalin
          - ES5340-Dig-In-HW::Es5340diginhw

Globals

Option	Value
Name	Es5340diginhw
Inverter Gate U High	Dig-In-HW-Channel #00
Inverter Gate U Low	Dig-In-HW-Channel #01
Inverter Gate V High	Dig-In-HW-Channel #02
Inverter Gate V Low	Dig-In-HW-Channel #03
Inverter Gate W High	Dig-In-HW-Channel #04
Inverter Gate W Low	Dig-In-HW-Channel #05
Inverter Auxiliary Trigger	---
Lower Threshold Dig-In-HW-Channel #00	1.5
Upper Threshold Dig-In-HW-Channel #00	3.5

6.6 User Interfaces

A GUI for LABCAR-EE is included with LABCAR-MODEL-PMSM. It allows to operate the model conveniently and contains layers for selecting various modes of operation, for entering the license key, and for obtaining the model status.

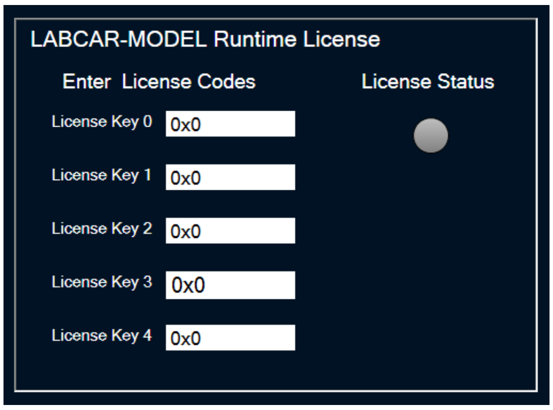
The following layers are available:

- PMSM Runtime License
- PMSM Simulation Control
- PMSM DAC
- PMSM Failure Simulations
- PMSM Parameters

6.6.1 PMSM Runtime License

This GUI provides an interface to enter license keys and to obtain the license status. The license keys are provided in hexadecimal notation.

The two model variants LABCAR-MODEL-PMSM Basic and LABCAR-MODEL-PMSM Advanced require different license keys. As a result, the signals "License Key 0-4" to be entered there are different.



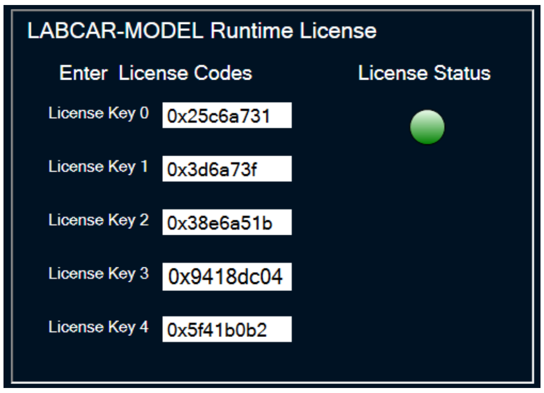
LABCAR-MODEL Runtime License

Enter License Codes

License Key 0	0x0
License Key 1	0x0
License Key 2	0x0
License Key 3	0x0
License Key 4	0x0

License Status

If you enter correct license keys/codes the "license status" will be green, otherwise it will remain gray.



LABCAR-MODEL Runtime License

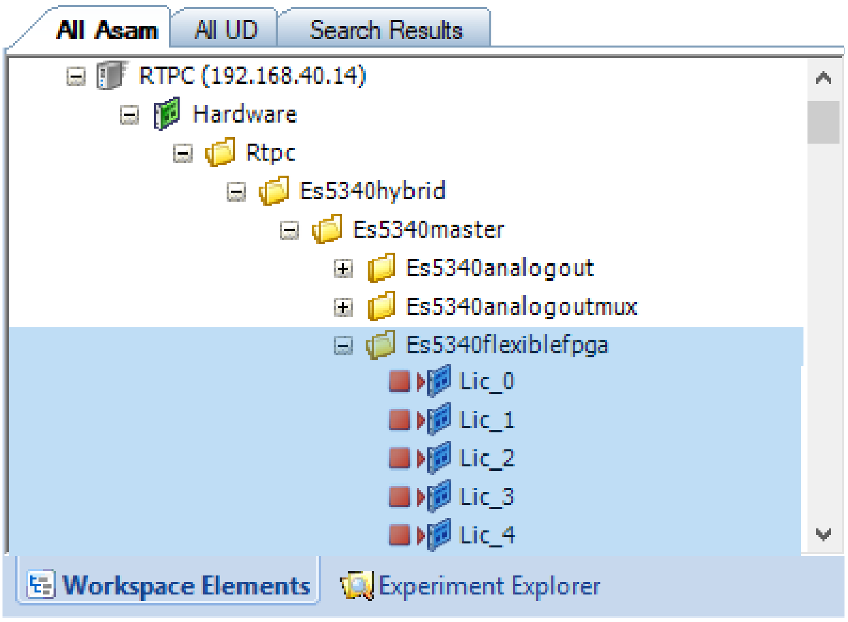
Enter License Codes		License Status
License Key 0	0x25c6a731	●
License Key 1	0x3d6a73f	
License Key 2	0x38e6a51b	
License Key 3	0x9418dc04	
License Key 4	0x5f41b0b2	

Entering the license keys manually to your project after every restart is a tedious task. For eliminating this effort, you may export the license keys as a partial parametrization of your project. LABCAR-EE offers a functionality to save and to load such a partial parametrization automatically after restart. It is described in detail in section 4.6.4 of the LABCAR-OPERATOR User's Guide [1].

Briefly, the steps for achieving an automatic loading of this partial parametrization are as follows.

To export the license keys as a partial parametrization

1. Download the experiment to your RTPC.
2. Enter the 5 license keys in LABCAR-EE.
3. Expand the HW ports with a right mouse button click in "Workspace Elements" and select "Show All HW Ports".
4. Navigate in the "Workspace Elements" to the previously entered license keys, i.e. to the labels mentioned below:

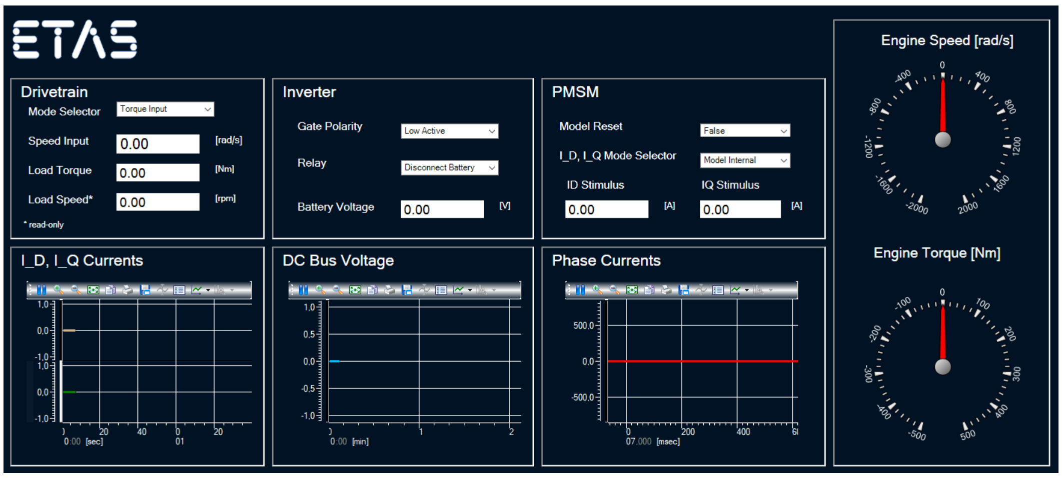


5. Select all 5 signals in the "Workspace elements".
6. Right click on the selected signals and select **Signal** → **Save Input Signal Settings...**.
7. Create a parameter file by selecting a file name.

8. Activate the "Add To Experiment" check box.
9. Activate the "Activate File" check box.
10. Select "Save".

6.6.2 PMSM Simulation Control

This is a GUI layer for the simulation control of LABCAR-MODEL-PMSM. The important interfaces for drivetrain, inverter and motor can be controlled using this layer.



Drivetrain
Mode Selector: Torque Input
Speed Input: 0.00 [rad/s]
Load Torque: 0.00 [Nm]
Load Speed*: 0.00 [rpm]
* read-only

Inverter
Gate Polarity: Low Active
Relay: Disconnect Battery
Battery Voltage: 0.00 [V]

PMSM
Model Reset: False
I_D, I_Q Mode Selector: Model Internal
ID Stimulus: 0.00 [A]
IQ Stimulus: 0.00 [A]

- "Drivetrain" Section of GUI

  Here you can enter the mechanical "Speed Input" (speed mode) and the "Load Torque" (torque mode).
- "Inverter" Section of GUI

  Here you enter the "Gate Polarity" of input PWM signals, the relay connection of the inverter with the battery, and the battery voltage.
- "PMSM" Section of GUI

  Here you enter the model reset and mode selector (model internal values/stimulation) with ID and IQ Stimulus input values.

NOTE

Use the mode selector to tune the resolver and DAC interfaces of the HiL setup.

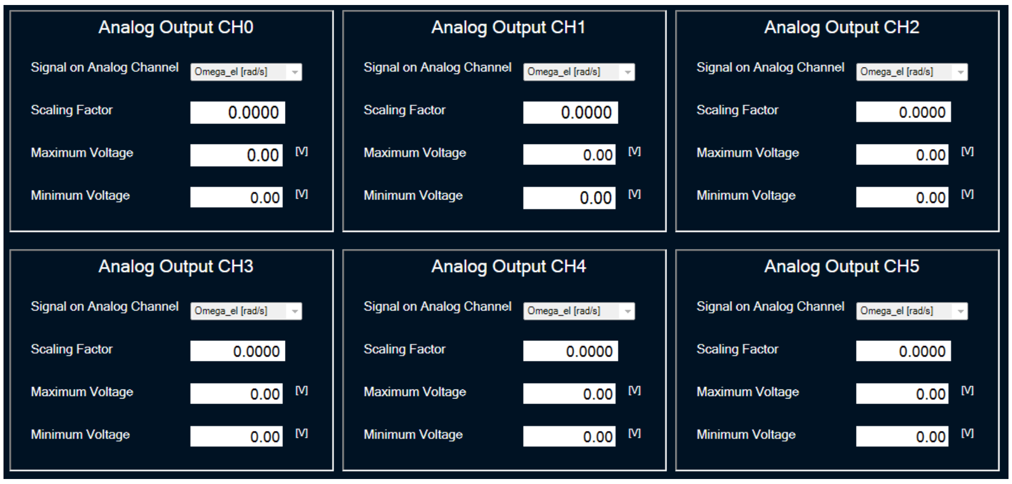
NOTE

The currents shown in the oscilloscope of the GUI have the unit [A], and the voltages have the unit [V].

6.6.3 PMSM DAC

As described in section "Signal Interface" on page 110, internal signals can be used with an analog output channel using a multiplexer. This GUI provides an easy interface to select signals and to input the scaling factor and minimum and maximum voltage values on the analog output channel.

Each analog output can be connected to one value of the simulation model. You can decide which signal is displayed, how this signal is scaled on the ±10 V output, if there is an offset to this signal, and what the maximum and minimum output voltages are (limited to ±10 V).



	Analog Output CH0	Analog Output CH1	Analog Output CH2	Analog Output CH3	Analog Output CH4	Analog Output CH5
Signal on Analog Channel	Omega_el [rad/s]	Omega_el [rad/s]	Omega_el [rad/s]	Omega_el [rad/s]	Omega_el [rad/s]	Omega_el [rad/s]
Scaling Factor	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Maximum Voltage	0.00 [V]	0.00 [V]	0.00 [V]	0.00 [V]	0.00 [V]	0.00 [V]
Minimum Voltage	0.00 [V]	0.00 [V]	0.00 [V]	0.00 [V]	0.00 [V]	0.00 [V]

For each signal used on the analog output channel, you can define a scaling factor using the parameter. The desired scaling factor can be computed by the following equation:

ScalingFactor = DesiredVoltageLevel / SignalValue

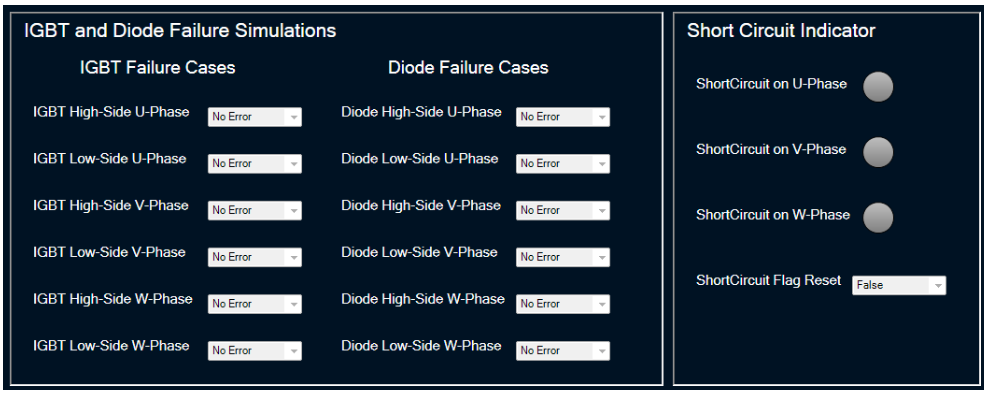
For example: To represent the model signal value of 200 A as an output voltage value of 10 V at the analog output of the ES5340.2-M, the scaling factor must be set to 10 V divided by 200 A = 0.05 V/A.

NOTE

To set the Offset to the signal voltage, use the model parameter "Offset_Anaout_X(X=0...6)".

6.6.4 PMSM Failure Simulations

This GUI layer provides an interface to insert various fault (stuck on "high" or stuck on "low") states to IGBT and diode. The short circuit indication flag on each phase is provided. A reset is provided to clear the short circuit flag.



IGBT and Diode Failure Simulations

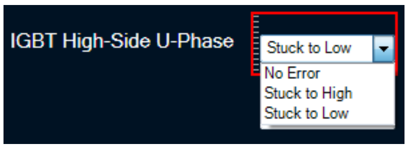
IGBT Failure Cases		Diode Failure Cases	
IGBT High-Side U-Phase	No Error	Diode High-Side U-Phase	No Error
IGBT Low-Side U-Phase	No Error	Diode Low-Side U-Phase	No Error
IGBT High-Side V-Phase	No Error	Diode High-Side V-Phase	No Error
IGBT Low-Side V-Phase	No Error	Diode Low-Side V-Phase	No Error
IGBT High-Side W-Phase	No Error	Diode High-Side W-Phase	No Error
IGBT Low-Side W-Phase	No Error	Diode Low-Side W-Phase	No Error

Short Circuit Indicator

ShortCircuit on U-Phase	●
ShortCircuit on V-Phase	●
ShortCircuit on W-Phase	●
ShortCircuit Flag Reset	False

- IGBT and Diode failure simulations

  For each switch/diode three states are possible.

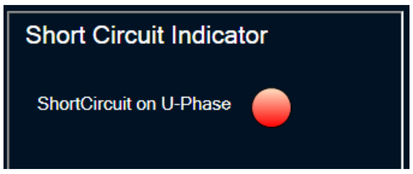
  - No Error:

    This is a normal working state of the switch/diode i.e. the input PWM signal is considered.
  - Stuck to High:

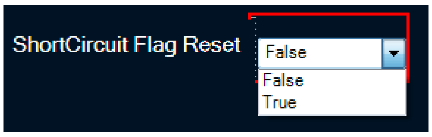
    The switch is stuck to high potential.
  - Stuck to Low:

    The switch is stuck to low potential.
- Short circuit indicator

  For each phase of the inverter, a flag indicates if a short occurs i.e. LED turns red if there is a short-circuit.

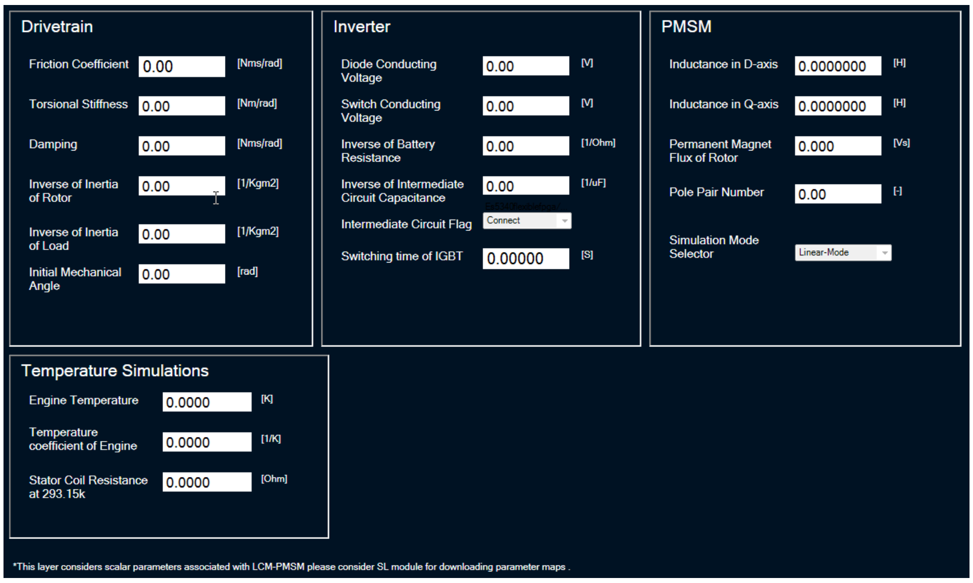
  The "Reset" input can be used to clear the short-circuit indication flags of all phases.

6.6.5 PMSM Parameters

This layer consists of an interface to scalar parameters of the drivetrain, the inverter, and the PMSM motor.

The interfaces of the subsystems, e.g. drivetrain, inverter, and PMSM motor, are documented in detail in the section “Signal Interface” on page 110.



Drivetrain

Friction Coefficient	0.00	[Nms/rad]
Torsional Stiffness	0.00	[Nm/rad]
Damping	0.00	[Nms/rad]
Inverse of Inertia of Rotor	0.00	[1/Kgm2]
Inverse of Inertia of Load	0.00	[1/Kgm2]
Initial Mechanical Angle	0.00	[rad]

Inverter

Diode Conducting Voltage	0.00	[V]
Switch Conducting Voltage	0.00	[V]
Inverse of Battery Resistance	0.00	[1/Ohm]
Inverse of Intermediate Circuit Capacitance	0.00	[1/uF]
Intermediate Circuit Flag	Connect	
Switching time of IGBT	0.00000	[S]

PMSM

Inductance in D-axis	0.0000000	[H]
Inductance in Q-axis	0.0000000	[H]
Permanent Magnet Flux of Rotor	0.000	[Vs]
Pole Pair Number	0.00	[-]
Simulation Mode Selector	Linear-Mode	

Temperature Simulations

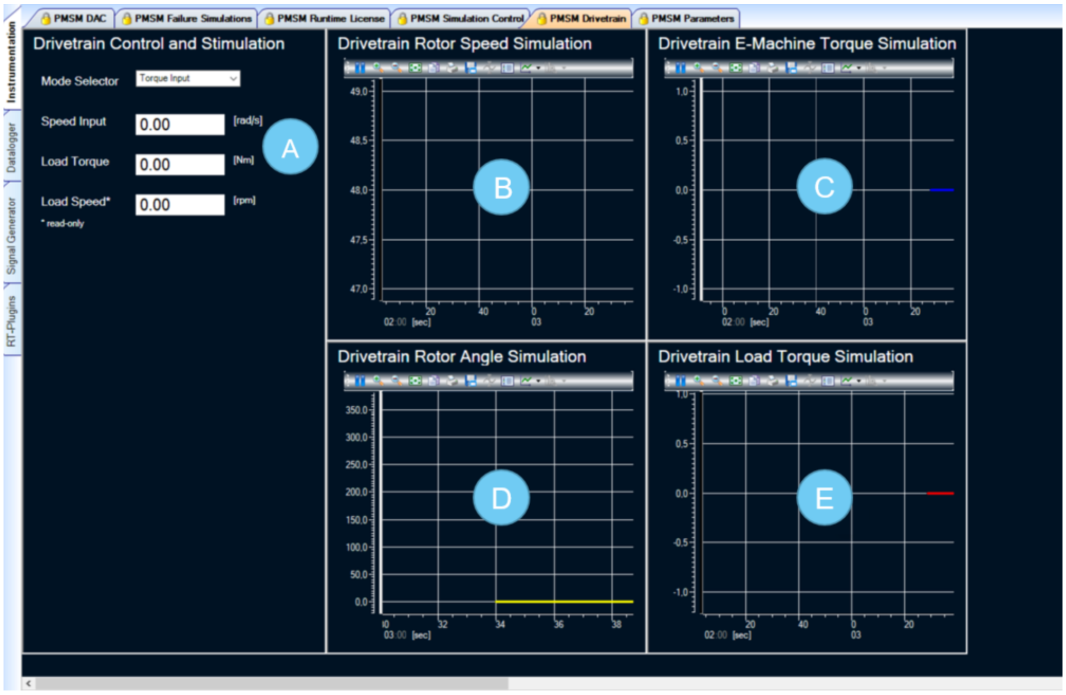
Engine Temperature	0.0000	[K]
Temperature coefficient of Engine	0.0000	[1/K]
Stator Coil Resistance at 293.15k	0.0000	[Ohm]

*This layer considers scalar parameters associated with LCM-PMSM please consider SL module for downloading parameter maps .

The GUI uses standard instruments. This can easily be used as a template for individual GUIs.

6.6.6 PMSM Drivetrain

The GUI provides information about the drivetrain including the output of the hardware item ES5340-RPM.



Drivetrain Control and Stimulation

Mode Selector	Torque Input	
Speed Input	0.00	[rad/s]
Load Torque	0.00	[Nm]
Load Speed*	0.00	[rpm]

* read-only

The GUI shows the following 5 sections:

A Torque or speed stimulation settings of drivetrain

B Model rotor speed information - n_rotor_rpm

C E-Machine torque information - mechanical as well as electrical

D Rotor angle information of the model and ES5340-RPM hardware item - phi_rotor_deg, ActMechAngle

E Model load torque information - load torque only

6.7 Migration from PMSM Model for ES5340 V1.0

Before the LABCAR-MODEL-PMSM V4.1.1 release, a former version existed with name FPGA Inverter/PMSM Model for ES5340 V1.0. Functionality and naming conventions changed from that version. This section serves as an entry point to users with the intent to migrate their project from the older model to the current LABCAR-MODEL-PMSM V4.x. It refers to the appropriate sections within this User's Guide and highlights differences between the V1.0 and V4.x.

If an existing project with the old version V1.0 shall be migrated to LABCAR-MODEL-PMSM V4.x, several measures have to be taken.

The following subsections provide you with hints and references for a successful migration of your old project.

6.7.1 FlexibleFPGA License

The LABCAR-MODEL-PMSM needs a FlexibleFPGA license tag, which is included in every delivery. The flashing of the FPGA, as described in the next section, can only take place, if this license has been installed into your ES5340 board. For installing such a license refer to "FlexibleFPGA V3.0.0 - Workflow Description: Tutorial" [4].

6.7.2 Flashing of FPGA

For a project containing LABCAR-MODEL-PMSM V4.x, it is necessary to flash the model to the ES5340 board. This procedure is explained in detail in section “Flashing Procedure” on page 89.

6.7.3 RTIO Modifications

The old version and LABCAR-MODEL-PMSM V4.x both offer the possibility to have certain model output signals assigned to hardware output pins.

However, the DAC option changed with the introduction of V4.x in various manners.

- At first, the DAC functionality is available for a different set of signals now. Some signals, which have been eligible for DAC in V1.0, are not eligible in V4.x anymore and vice versa. For a comprehensive overview about the changes in the DAC functionality and other changes to the input and output signals, refer to the tables in section “Parameters and Interfaces” on page 129.
- At second, the way a DAC output is configured changed. The new procedure to configure LABCAR-MODEL-PMSM DAC signals are described in section “Analog Outputs” on page 119 and “PMSM DAC” on page 124. A list of output signals that changed from V1.0 to V4.x is given below in section “Parameters and Interfaces” on page 129.

- With V1.0 the mapping of the signal onto the output pin was done by providing minimum and maximum values for the signal and the output voltage, whereby a scaling factor was calculated internally.

  Now with V4.x you provide several factors to the system, which are used to determine the output pin voltage. These factors are:
  - a scaling factor
  - an offset
  - maximal output boundaries
  - minimal output boundaries
- While with version V1.0, the RTIO configuration was immediately available in the project, LABCAR-MODEL-PMSM V4.x requires the loading of the corresponding ADF-file. How to do this is described in section “Project Creation in LABCAR-OPERATOR” on page 91.

6.7.4 License Information

The license information is provided in LABCAR-MODEL-PMSM V4.x via a license key interface comprising 5 license keys. An indication whether the license keys are valid is returned via a boolean output signal.

The license key interface is described in section “License Keys Interface” on page 119.

6.7.5 Parameters and Interfaces

The following tables list all inputs, outputs and parameters of both model versions and highlights differences. Differences occur in existence, naming and placement. Also *false friends* exist, whereby variables with identical or very similar names can be found in both model versions but do not constitute the same parameter with respect to its semantic meaning.

6.7.5.1 Drivetrain

Inputs

Description	Name (unit) V1.0	Name (unit) V4.x	Difference
Load torque	T_L (Nm)	T_L (Nm)	-
Initial mechanical rotor angle	Phi_ mech_Init (°)	Phi_mech_Init (rad)	-
Speed input of drivetrain in speed mode	Omega_ mech_init (1/s)	Omega_Input (rad/s)	-
Generated electrical engine torque	-	Torque (Nm)	labeled as output of PMSM in V4.x

Outputs

Description	Name (unit) V1.0	DAC	Name (unit) V4.x	DAC	Difference
Mechanical rotor speed	Omega_ mech (1/s)	yes	Omega_ mech (rad/s)	yes	-
Electrical rotor speed	Omega_el (°)	yes	Omega_el (rad/ s)	yes	-
Mechanical rotor angle	Phi_mech (°)	yes	Phi_ mech_rad (rad)	yes	-
Electrical rotor angle	Phi_el (°)	yes	Phi_el (rad)	yes	-
Resulting mechanical rotor torque	T_mech (Nm)	yes	T_mech (Nm)	yes	-
Average electrical torque	-	-	Tel_average (Nm)	-	non existent in V1.0

Parameters

Description	Name (unit) V1.0	Name (unit) V4.x	Difference
Friction coefficient of E-machine	f (Nms/rad)	f (Nms/rad)	-
Torsional stiffness between rotor and load	C (Nm/rad)	C (Nm/rad)	-
Damping between rotor and load	D (Nms/rad)	D (Nms/rad)	-
Mode selector 0: use torque input 1: use speed input	-	Direct Feed through	non existent in V1.0
Inertia of rotor	J_E (kgm^2)	invJ_E (1/kgm^2)	In V4.x the inverse is given
Inertia of load/drive-train	J_L (kgm^2)	invJ_L (1/kgm^2)	In V4.x the inverse is given

6.7.5.2 Inverter

Inputs

Description	Name (unit) V1.0	Name (unit) V4.x	Difference
Input voltage of battery	U_Batt0 (V)	U_Batt0 (V)	-
Intermediate circuit disable flag 0: Intermediate circuit enabled 1: Intermediate circuit disabled	-	RC_Ckt_Disable_Flag	non existent in V1.0
Switch to connect/disconnect battery 0: Battery disconnected 1: Battery connected	-	Relay_Switch	non existent in V1.0
Input for resetting short circuit error flags	-	ShortCircuit_Flag_Reset	non existent in V1.0
6 signals for IGBT control	-	PWM_*	
6 signals for diode failure	-	Fail_Diode_*	non existent in V1.0
IGBT Polarity Control 0: Low active, conducting while gate is "Low" 1: High active, conducting while gate is "High"	Gate_Signal_Polarity	Gate_Signal_Polarity	-
6 signals for IGBT failure	Fail_HS_X, Fail_LS_X (X=U,V,W)	Fail_IGBT_*	-

Outputs

Description	Name (unit) V1.0	DAC V1.0	Name (unit) V4.x	DAC V4.x	Difference
Intermediate circuit voltage	U_CDCneu (V)	Yes	U_DC (V)	Yes	-
Battery current	I_Batt (A)	Yes	I_Bat (A)	No	-
Intermediate circuit current	I_DC (A)	Yes	I_DC (A)	Yes	-
Voltage phase U	u_phase (V)	Yes	U_phase_U (V)	Yes	-
Voltage phase V	u_phase (V)	Yes	U_phase_V (V)	Yes	-
Voltage phase W	u_phase (V)	Yes	U_phase_W (V)	Yes	-
Current through upper switch phase U	i_element (A)	Yes	I_element_H_U (A)	No	-
Current through upper switch phase V	i_element (A)	Yes	I_element_H_V (A)	No	-
Current through upper switch phase W	i_element (A)	Yes	I_element_H_W (A)	No	-
Current through lower switch phase U	i_element (A)	Yes	I_element_L_U (A)	No	-
Current through lower switch phase V	i_element (A)	Yes	I_element_L_V (A)	No	-
Current through lower switch phase W	i_element (A)	Yes	I_element_L_W (A)	No	-
Voltage on upper switch phase U	u_switch (V)	Yes	U_Switch_H_U (V)	No	-
Voltage on upper switch phase V	u_switch (V)	Yes	U_Switch_H_V (V)	No	-
Voltage on upper switch phase W	u_switch (V)	Yes	U_Switch_H_W (V)	No	-
Voltage on lower switch phase U	u_switch (V)	Yes	U_Switch_L_U (V)	No	-
Voltage on lower switch phase V	u_switch (V)	Yes	U_Switch_L_V (V)	No	-
Voltage on lower switch phase W	u_switch (V)	Yes	U_Switch_L_W (V)	No	-
Flag set if both upper and lower switches of phase U are open	-	-	Blocked_U	-	non existent in V1.0
Flag set if both upper and lower switches of phase V are open	-	-	Blocked_V	-	non existent in V1.0
Flag set if both upper and lower switches of phase W are open	-	-	Blocked_W	-	non existent in V1.0

Parameters

Description	Name (unit) V1.0	Name (unit) V4.x	Difference
Conducting voltage in diode (293.15K)	U_D_Fwd (V)	U_D_Fwd (V: 0.01...3)	-
Negative of U_T0_Diode	-	-	non existent in V4.x
Inner resistance of battery	R_Batt (Ω)	Inv_R_Batt (1/ Ω : 1e-3...10)	Inverted in V4.x
Inverse of intermediate circuit capacitor	C_DC [mF]	Inv_C_DC (1/μF: 1e-3...1e-1)	Inverted in V4.x
Switching time of IGBT	T_Switch [s]	T_Switch (s: 0.2e-6...10e6)	-
Conducting resistance in IGBT	R_SW_ON (Ω)	R_SW_ON (Ω : 0.00...0.05)	-
Conducting resistance in diode	R-D_Fwd (Ω)	R_D_FWD (Ω : 0.00...0.05)	-
Conducting voltage in IGBT(293.15K)	U_SW_Fwd (V)	U_SW_Fwd (V: 0.01...3)	-

6.7.5.3 PMSM

Inputs

Description	Name (unit) V1.0	Name (unit) V4.x	Difference
Stator coil resistance	Rs	Rs (Ω)	-
Resistance represents iron losses of machine	-	Riron (Ω)	non existent in V1.0
Inductance in d-axis	Ld (H)	L_d (H)	-
Inductance in q-axis	Lq (H)	L_q (H)	-
Permanent magnetic flux in rotor	Psi (Vs)	Psi_Rotor (Vs)	-
Current DQ Selector 0: Internally calculated values for I_d and I_q are used. 1: Stimulus inputs "ID_Stimulate" and "IQ_Stimulate" are used.	-	Current_DQ_Selector	non existent in V1.0
Stimulating input for I_d	Ia_init (A)	ID_Stimulate (A)	-
Stimulating input for I_q	Ib_init (A)	IQ_Stimulate (A)	-
Reset all internal states	-	Reset	non existent in V1.0
Number of samples considered for angle correction of I_d and I_q calculations	-	AngleCorrectionSamples	non existent in V1.0
Rotor electrical angular frequency	ω (1/s)	Omega_el (rad/s)	-
Electrical rotor angle	X (rad)	Phi_el (rad)	-
3 strand failure inputs	strang_fehler	-	non existent in V4.x
3 phase voltage inputs	U_{phase} (V)	-	non existent in V4.x

Outputs

Description	Name (unit) V1.0	DAC V1.0	Name (unit) V4.x	DAC V4.x	Difference
Voltage in d-axis	-	-	Vd (V)	-	not provided by V1.0
Voltage in q-axis	-	-	Vq (V)	-	not provided by V1.0
Flux in d-axis	-	-	Psi_d (Vs)	-	not provided by V1.0
Flux in q-axis	-	-	Psi_q (Vs)	-	not provided by V1.0
Motor electromagnetic torque	T_{el} (Nm)	Yes	Torque (Nm)	No	-
Induced back EMF of phase U	U_P (V)	Yes	U_Ind_U (V)	Yes	-
Induced back EMF of phase V		Yes	U_ind_V (V)	Yes	-
Induced back EMF of phase W		Yes	U_ind_W (V)	Yes	-
Current of phase U	I_{phase} (A)	Yes	I_Phase_U (A)	Yes	-
Current of phase V	-	Yes	I_Phase_V (A)	Yes	-
Current of phase W	-	Yes	I_Phase_W (A)	Yes	-
Current in d-axis	I_d (A)	-	Id (A)	-	-
Current in q-axis	I_q (A)	-	Iq (A)	-	-
Copper loss of motor	P_loss_M	-	PLoss_Copper (W)	-	V1.0 only provides overall loss value
Core loss of motor	-	-	PLoss_Core (W)	-	-
Current in core loss branch of d-axis	-	-	Icd (A)	-	not provided by V1.0
Current in core loss branch q-axis	-	-	Icq (A)	-	not provided by V1.0
Current in magnetization branch of d-axis	-	-	Imd (A)	-	not provided by V1.0
Current in magnetization branch of q-axis	-	-	Imq (A)	-	not provided by V1.0
Zero position (digital)	Zero_Pos	-	-	-	not provided by V4.x
Failure signal 1: overflow of fk addition 0: otherwise	Failure_OF	-	-	-	Please contact technical support if you need assistance

Parameters

Description	Name (Unit) V1.0	Name (range) V4.x	Difference
Number of pole pairs	p	PPN (1...32)	-
Peak value of cogging torque	Tcm (Nm)	-	Cogging torque is not implemented in V4.x

7 LABCAR-MODEL-FC

ETAS LABCAR-MODEL-FC is a detailed model of a fuel-cell (FC) system, which provides components to simulate the fuel-cell system in a fuel-cell solution. A typical fuel-cell system consists of the following sub systems:

- Cathode Supply System
- Hydrogen Supply System
- Coolant Supply System
- Stack

You find an introduction to each sub system in “Fuel-Cell Subsystems and Components” on page 155.

ETAS LABCAR-MODEL-FC is implemented in MATLAB/Simulink using S-Functions. No additional toolboxes are required for the model execution. LABCAR-MODEL-FC is targeting towards the automotive domain. The model is realtime-capable and targeted for the execution on an ETAS LABCAR HiL system. Execution on other platforms may work as well. LABCAR-MODEL-FC includes LABCAR-EE layers, so that a basic GUI is available.

7.1 Project and Experiment Creation

In the following section, the step-by-step guide describes the generation of a COSYM and LABCAR-OPERATOR project using LABCAR-MODEL-FC. For the execution of the project, no physical ECU is needed. Instead, a SoftFCCU controls the fuel-cell system model. For providing the electrical and thermal power requirements to the SoftFCCU, a basic restbus model of a vehicle is provided.

Please note that the SoftFCCU is a simplistic model prepared only for demonstrating the behavior of the FC system by setting up control loops between FC sub systems (Cathode, Anode, Coolant) and the SoftFCCU.

7.1.1 Project Creation in COSYM

This section addresses qualified personnel with knowledge in using COSYM. If you do not have any experience in using COSYM, please read first [6].

For the manual project creation using LABCAR-MODEL-FC with COSYM follow the steps described in this section.

To set up a COSYM project

1. Open the COSYM application.

   The COSYM application can already have a project open, in this case
2. Click **File** → **New** → **Project.**

or

   If no previous project is pre-loaded,

   - Click the icon "Create new project".
3. Choose the location for the project and name the project.
4. Click **Create.**

NOTE

Refer to section 7.1 in [6] for additional help regarding the project creation in COSYM.

> The "Library" section of the project now is empty to start with. A default system is created with the name "System" and is automatically mapped to "HiL_Target".

To add a hardware module to a system

1. Open the "System" from the "Systems" pane by double click.
2. Expand "System".
3. Right click on "HiL_Target" and select "Create RTIO module" from the drop-down menu.

> The "System" shows the added hardware module in the "Diagram" view.

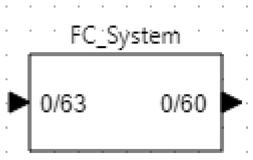
The fuel-cell system model consists of a single MATLAB/Simulink module. This model comes with a SoftFCCU, which mimics very basic behavior of an FCCU to get the fuel-cell system into operational mode.

To add a Simulink model for the FC_System

1. In the Library pane, right click on the "Control" folder and choose "Import Model".
2. Under "Source of the model to be imported", click on the **Browse** button.
3. Select the LMD file `FC_System.lmd` which internally refers to the model configuration `FC_System.mdl`. The corresponding LMD file `FC_System.lmd` is located in the installation folder:
   `Project Items\COSYM\FC_Default_StackSoftFCCU\mdl\Simulink_FC_System\src\FC_System_LABCAR_rtw\`
4. Click the **Import** button.

   The model "FC_System" is added to the "Library" under the "Control" folder.
5. Drag and drop the "FC_System" model from the "Library" pane
   - to the "Diagram" area of the "System".
   - to the "Units" folder of "System" under the "Systems" pane.

> The "System" shows the added module in the "Diagram" view.



NOTE

The Simulink model references additional dependencies by using the "`ExternalFiles.xml`". Please see section “Mapping External Files for HiL Execution” on page 18 for further details.

The fuel-cell system model requires initial environment and gas conditions for proper simulation. For this purpose, C-Code modules are used for temperature, pressure and gas components initialization.

The following C-Code modules have to be added:

- `InitializationTemperature.lmd`

  Located in the installation folder at
  `Project Items\COSYM\FC_Default_StackSoftFCCU\cmod\InitializationTemperature\`
- `InitializationPressure.lmd`

  Located in the installation folder at
  `Project Items\COSYM\FC_Default_StackSoftFCCU\cmod\InitializationPressure\`
- `InitializationGasComponent1.lmd`

  Located in the installation folder at
  `Project Items\COSYM\FC_Default_StackSoftFCCU\cmod\InitializationGasComponent1\`
- `InitializationGasComponent2.lmd`

  Located in the installation folder at
  `Project Items\COSYM\FC_Default_StackSoftFCCU\cmod\InitializationGasComponent2\`
- `InitializationGasComponent3.lmd`

  Located in the installation folder at
  `Project Items\COSYM\FC_Default_StackSoftFCCU\cmod\InitializationGasComponent3\`

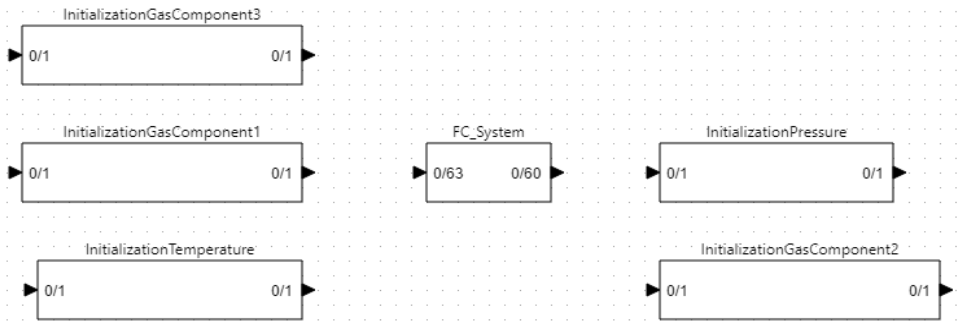
To add a C-Code module

1. In the Library pane, right click on the "Environment" folder and choose "Import Model".
2. Under "Source of the model to be imported", click on the **Browse** button.
3. Select the C-Code module's LMD file from the locations specified above.
4. Click on the **Import** button.
5. The C-Code model is added to the "Library" under the "'Environment" folder.
6. Open the "System" from the "Systems" pane by double click.
7. Drag and drop the C-Code model from the "Library" pane
   - to the Diagram area of the "System".

- to the "Units" folder of "System" under the "Systems" pane.

8. Repeat the steps 1 to 7 for the 5 C-Code modules listed above.

> The "System" shows the added C-Code module in the "Diagram" view.


InitializationGasComponent3
0/1
0/1
InitializationGasComponent1
0/1
0/1
FC_System
0/63
0/60
InitializationPressure
0/1
0/1
InitializationTemperature
0/1
0/1
InitializationGasComponent2
0/1
0/1

By default, no model connections are established. For example, the SoftFCCU is not connected to the restbus or fuel-cell system model. Pre-defined connection files couple the restbus, SoftFCCU and fuel-cell system.

The following connection files can be added:

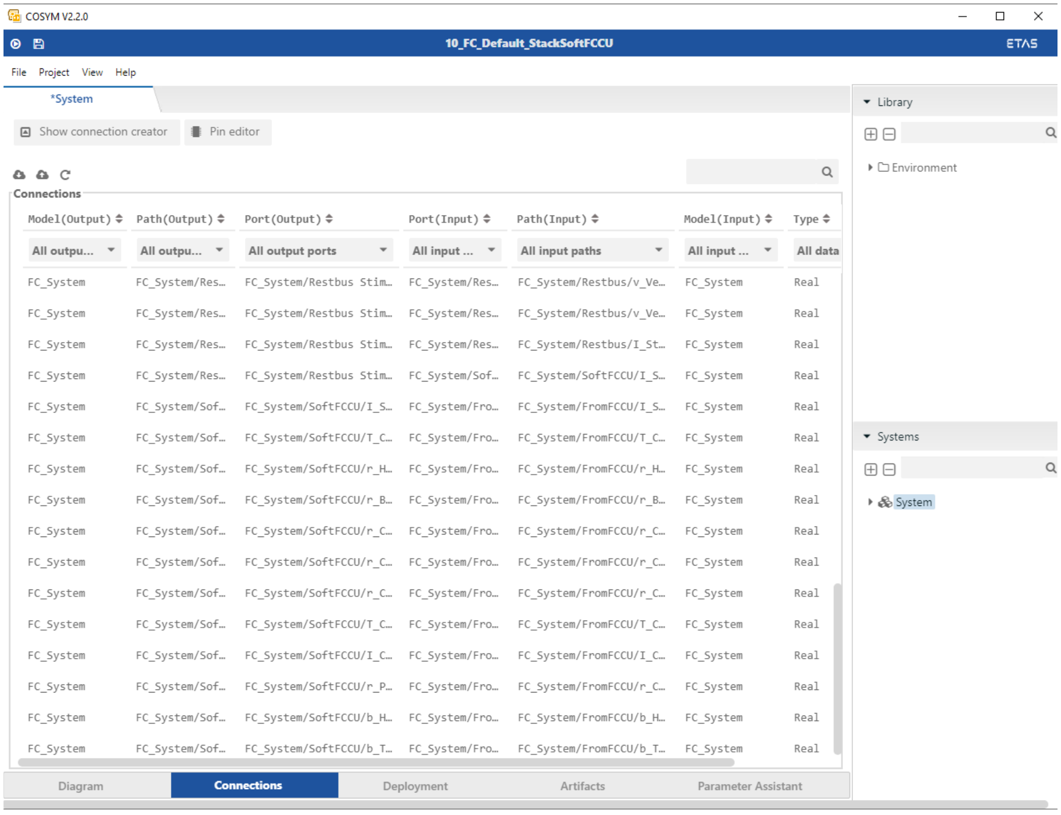
- `FC_Default__SoftFCCU.xml`
- `Restbus__FC_Default_viaInitialValues.xml`
- `Restbus__SoftFCCU.xml`
- `SoftFCCU__FC_Default.xml`

These files are located in the installation folder at
`Project Items\COSYM\FC_Default_StackSoftFCCU\conn`

To add connection files

1. Open the "Connections" view available at the bottom of the opened "System".
2. Click on the "Import connections" icon.
3. Select each XML file mentioned above.
4. Click on "Import" to import connections from each file.
5. Repeat the steps 1 to 4 for all the connection files mentioned above.

> After the connection files are added, the necessary connections are established between the Fuel-Cell system, SoftFCCU, and restbus. They are visible in the "Connections" view.



Connections

Model(Output)	Path(Output)	Port(Output)	Port(Input)	Path(Input)	Model(Input)	Type
All outpu...	All outpu...	All output ports	All input ...	All input paths	All input ...	All data
FC_System	FC_System/Res…	FC_System/Restbus Stim…	FC_System/Res…	FC_System/Restbus/v_Ve…	FC_System	Real
FC_System	FC_System/Res…	FC_System/Restbus Stim…	FC_System/Res…	FC_System/Restbus/v_Ve…	FC_System	Real
FC_System	FC_System/Res…	FC_System/Restbus Stim…	FC_System/Res…	FC_System/Restbus/I_St…	FC_System	Real
FC_System	FC_System/Res…	FC_System/Restbus Stim…	FC_System/Sof…	FC_System/SoftFCCU/I_S…	FC_System	Real
FC_System	FC_System/Sof…	FC_System/SoftFCCU/I_S…	FC_System/Fro…	FC_System/FromFCCU/I_S…	FC_System	Real
FC_System	FC_System/Sof…	FC_System/SoftFCCU/T_C…	FC_System/Fro…	FC_System/FromFCCU/T_C…	FC_System	Real
FC_System	FC_System/Sof…	FC_System/SoftFCCU/r_H…	FC_System/Fro…	FC_System/FromFCCU/r_H…	FC_System	Real
FC_System	FC_System/Sof…	FC_System/SoftFCCU/r_B…	FC_System/Fro…	FC_System/FromFCCU/r_B…	FC_System	Real
FC_System	FC_System/Sof…	FC_System/SoftFCCU/r_C…	FC_System/Fro…	FC_System/FromFCCU/r_C…	FC_System	Real
FC_System	FC_System/Sof…	FC_System/SoftFCCU/r_C…	FC_System/Fro…	FC_System/FromFCCU/r_C…	FC_System	Real
FC_System	FC_System/Sof…	FC_System/SoftFCCU/r_C…	FC_System/Fro…	FC_System/FromFCCU/r_C…	FC_System	Real
FC_System	FC_System/Sof…	FC_System/SoftFCCU/T_C…	FC_System/Fro…	FC_System/FromFCCU/T_C…	FC_System	Real
FC_System	FC_System/Sof…	FC_System/SoftFCCU/I_C…	FC_System/Fro…	FC_System/FromFCCU/I_C…	FC_System	Real
FC_System	FC_System/Sof…	FC_System/SoftFCCU/r_P…	FC_System/Fro…	FC_System/FromFCCU/r_C…	FC_System	Real
FC_System	FC_System/Sof…	FC_System/SoftFCCU/b_H…	FC_System/Fro…	FC_System/FromFCCU/b_H…	FC_System	Real
FC_System	FC_System/Sof…	FC_System/SoftFCCU/b_T…	FC_System/Fro…	FC_System/FromFCCU/b_T…	FC_System	Real

Fig. 7-1 COSYM Connections View for FC-System

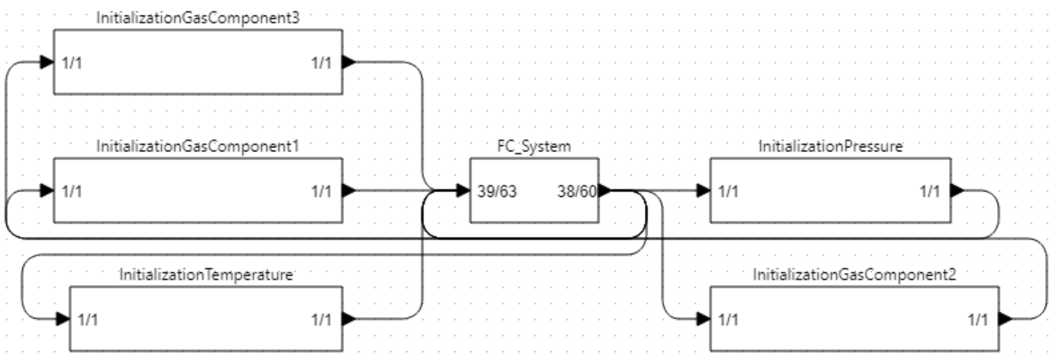

InitializationGasComponent3
1/1
1/1
InitializationGasComponent1
1/1
1/1
FC_System
39/63
38/60
InitializationPressure
1/1
1/1
InitializationTemperature
1/1
1/1
InitializationGasComponent2
1/1
1/1

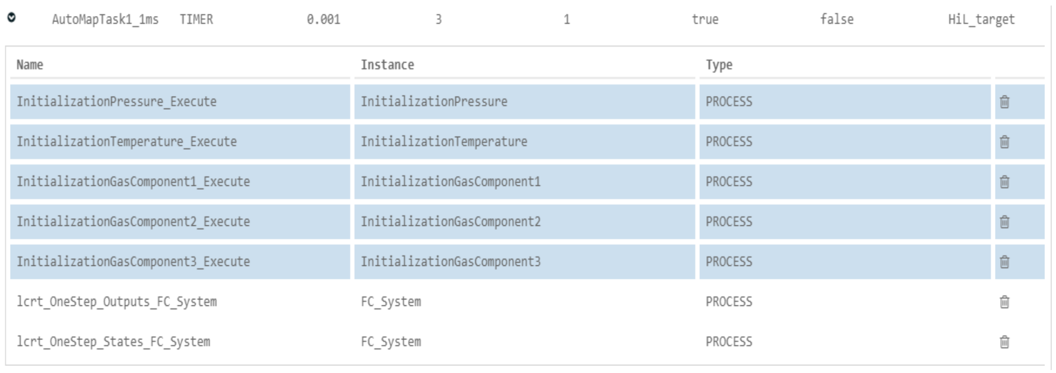
Fig. 7-2 COSYM Diagram View with Connections for FC-System

NOTE

Check each connection file and ensure that you added all connection files. Otherwise, the COSYM project might not work properly.

OS Configuration

Via the "Deployment" view, you can organize the processes of COSYM modules. For Simulink modules, the order of execution and the selection of a proper step size is important. By default, tasks are *not* auto-generated for Simulink modules when adding them to the project. Please click on **+ Auto mapping** to automatically map the processes to tasks. For added C-Code modules, ensure that the timer processes under tasks are executed before the Simulink model processes as shown in Fig. 7-3.



```
AutoMapTask1_1ms   TIMER   0.001   3   1   true   false   HiL_target
```

Name	Instance	Type
InitializationPressure_Execute	InitializationPressure	PROCESS
InitializationTemperature_Execute	InitializationTemperature	PROCESS
InitializationGasComponent1_Execute	InitializationGasComponent1	PROCESS
InitializationGasComponent2_Execute	InitializationGasComponent2	PROCESS
InitializationGasComponent3_Execute	InitializationGasComponent3	PROCESS
lcrt_OneStep_Outputs_FC_System	FC_System	PROCESS
lcrt_OneStep_States_FC_System	FC_System	PROCESS

Fig. 7-3 Execution of Timer Processes

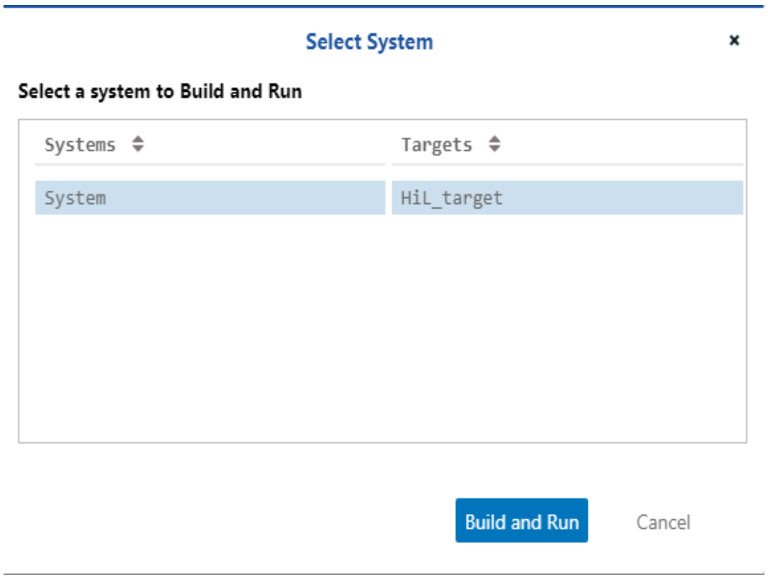
NOTE

Please refer to [6] section "OS Configuration" - "Ordering of the Tasks" to get familiar with the ordering of processes.

All relevant modules, necessary connections, and the OS configuration are now available and you can build a COSYM project.

To build a COSYM project

1. Choose "Project" → "Build and Run".
2. Click **Build and Run.**



Select a system to Build and Run

Systems	Targets
System	HiL_target

For more details about the Simulink code generation, refer to the section “Simulink Model - Code Generation Process” on page 14.

7.1.2 Project Creation in LABCAR-OPERATOR

This section addresses qualified personnel with knowledge in using LABCAR-OPERATOR. If you do not have any experience in using LABCAR-OPERATOR, please read first [1].

For the manual project creation using LABCAR-MODEL-FC with LABCAR-OPERATOR follow the steps described in this section.

To set up the LABCAR-OPERATOR project

1. Open the LABCAR-IP application.
2. Click **File** → **New Project**.
3. Choose the location for the file and name the project.
4. Use "RTPC" as "Target Name".
5. Click **Finish**.

> The Project Explorer contains an empty hardware configuration.

The fuel-cell system model consists of a single MATLAB/Simulink module. This model includes a SoftFCCU, which very basically mimics the behavior of an FCCU in order to bring the fuel-cell system into operational mode.

To add a Simulink model for FC_System

1. Choose **Project** → **Add Module.**
2. Select "Add Simulink (TM) Module".
3. Select "Use existing Simulink (TM) Model (will be copied)".

4. Select the fuel-cell system model configuration FC_System. The corresponding model `FC_System.mdl` is located in the install folder

   Project Items\LCO\FC_Default_StackSoftFCCU\mdl\Simulink_FC_System\src
5. Activate the checkbox "Copy Model Directory into Project".
6. Click **Finish**.

> The Project Explorer shows the added model.

The fuel-cell system model requires initial environment and gas conditions for a proper simulation. For this purpose, C-Code modules are used for temperature, pressure and gas components initialization.

NOTE

The Simulink model references additional dependencies by using the "`ExternalFiles.xml`". Please see section “Mapping External Files for HiL Execution” on page 18 for further details.

The following C-Code modules have to be added:

- InitializationTemperature.lmd

  Located in the installation folder at
  `Project Items\LCO\FC_Default_StackSoftFCCU\cmod\InitializationTemperature\`
- InitializationPressure.lmd

  Located in the installation folder at
  `Project Items\LCO\FC_Default_StackSoftFCCU\cmod\InitializationPressure\`
- InitializationGasComponent1.lmd

  Located in the installation folder at
  `Project Items\LCO\FC_Default_StackSoftFCCU\cmod\InitializationGasComponent1\`
- InitializationGasComponent2.lmd

  Located in the installation folder at
  `Project Items\LCO\FC_Default_StackSoftFCCU\cmod\InitializationGasComponent2\`
- InitializationGasComponent3.lmd

  Located in the installation folder at
  `Project Items\LCO\FC_Default_StackSoftFCCU\cmod\InitializationGasComponent3\`

To add a C-Code module

1. Choose **Project** → **Add Module**.
2. Select "Add C-Code Module".
3. Select "Use existing Module (will be copied)".
4. Select one of the C-Code modules listed above.
5. Click **Finish**.

6. Repeat the steps 1 to 5 for all the five C-Code modules listed above.

> The Project Explorer shows the added C-Code modules.

By default, no model connections are established. For example, the SoftFCCU is not connected to the restbus or fuel-cell system model. Pre-defined connection files couple the restbus, SoftFCCU and fuel-cell system.

The following connection files can be added:

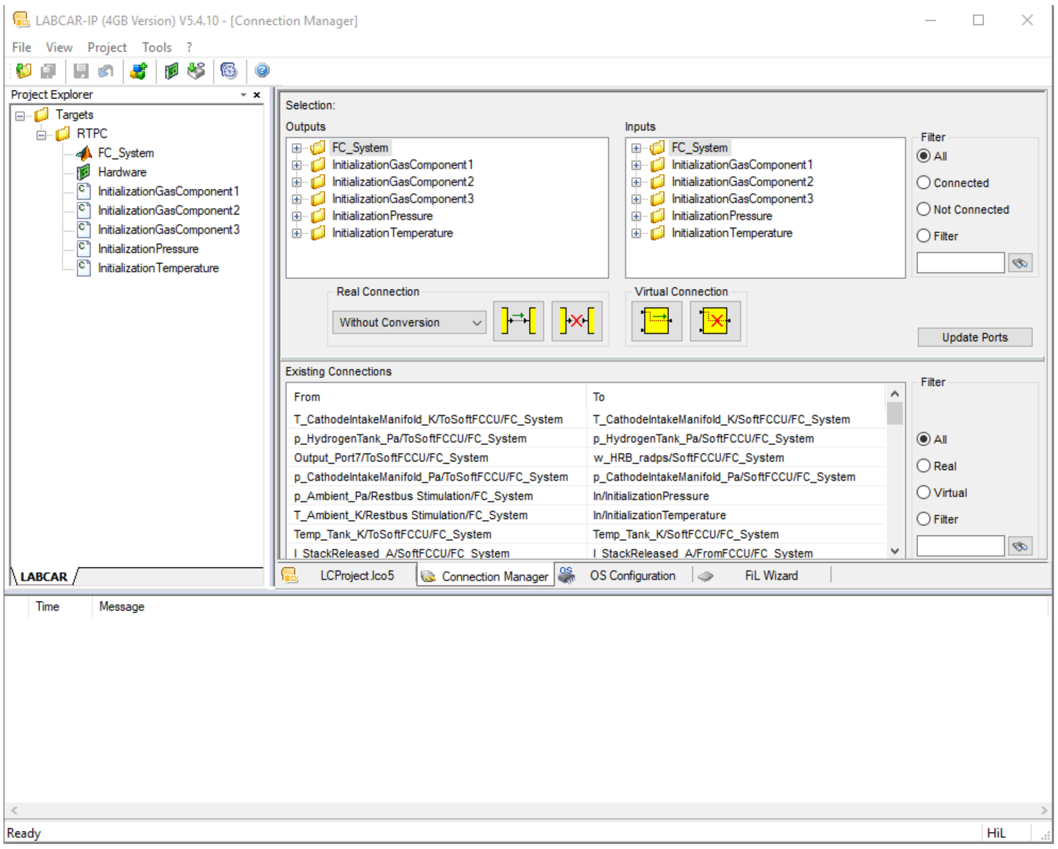
- FC_Default__SoftFCCU.xml
- Restbus__FC_Default_viaInitial_Values.xml
- Restbus__SoftFCCU.xml
- SoftFCCU__FC_Default.xml

These files are located in the installation folder at
`Project Items\LCO\FC_Default_StackSoftFCCU\conn`

To add connection files

1. Open the "Connection Manager" and right-click in the area "Existing Connections".
2. Select "Import Connections..." from the drop-down menu.
3. Select each connection file mentioned above.
4. Click on "Open" to import connections from each file.
5. Repeat the steps 1 to 4 for all the connection files mentioned above.

> After the connection files are added, the necessary connections are established between the fuel-cell system, SoftFCCU, and restbus and they are visible in the "Connection Manager".



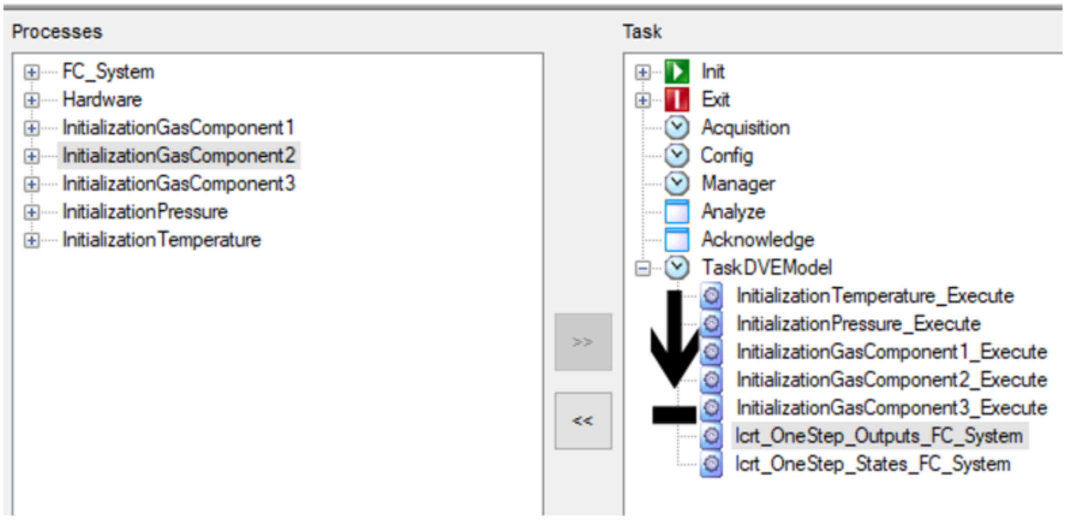
From	To
T_CathodeIntakeManifold_K/ToSoftFCCU/FC_System	T_CathodeIntakeManifold_K/SoftFCCU/FC_System
p_HydrogenTank_Pa/ToSoftFCCU/FC_System	p_HydrogenTank_Pa/SoftFCCU/FC_System
Output_Port7/ToSoftFCCU/FC_System	w_HRB_radps/SoftFCCU/FC_System
p_CathodeIntakeManifold_Pa/ToSoftFCCU/FC_System	p_CathodeIntakeManifold_Pa/SoftFCCU/FC_System
p_Ambient_Pa/Restbus Stimulation/FC_System	In/InitializationPressure
T_Ambient_K/Restbus Stimulation/FC_System	In/InitializationTemperature
Temp_Tank_K/ToSoftFCCU/FC_System	Temp_Tank_K/SoftFCCU/FC_System
I_StackReleased_A/SoftFCCU/FC_System	I_StackReleased_A/FromFCCU/FC_System

NOTE

Check each connection file and ensure that you added all connection files. Otherwise, the project might not work properly.

OS configuration

Via the "OS Configuration" tab, you can organize the processes of LABCAR-OPERATOR modules. For Simulink modules, the order of execution and the selection of a proper step size is important. By default, proper tasks for the Simulink model are auto-generated when adding the module to the project. For added C-Code modules you have to assign the execute processes manually to ensure that the timer processes under tasks are executed before the Simulink model processes as shown below. Please ensure that "Init" processes are assigned to the "Init" task. In most cases, this happens automatically.


Processes
FC_System
Hardware
InitializationGasComponent1
InitializationGasComponent2
InitializationGasComponent3
InitializationPressure
InitializationTemperature
>>
<<
Task
Init
Exit
Acquisition
Config
Manager
Analyze
Acknowledge
TaskDVEModel
InitializationTemperature_Execute
InitializationPressure_Execute
InitializationGasComponent1_Execute
InitializationGasComponent2_Execute
InitializationGasComponent3_Execute
lcrt_OneStep_Outputs_FC_System
lcrt_OneStep_States_FC_System

NOTE

Please refer to LABCAR-OPERATOR User's Guide [1] section 3.15.2 "Working with OS Configurations" to get familiar with the ordering of processes.

All relevant modules, necessary connections, and the OS configuration are now available.

As a next step, you can build the LABCAR-OPERATOR project.

NOTE

Be sure to select the proper MATLAB version using the MATLAB Version Selector. See [2].

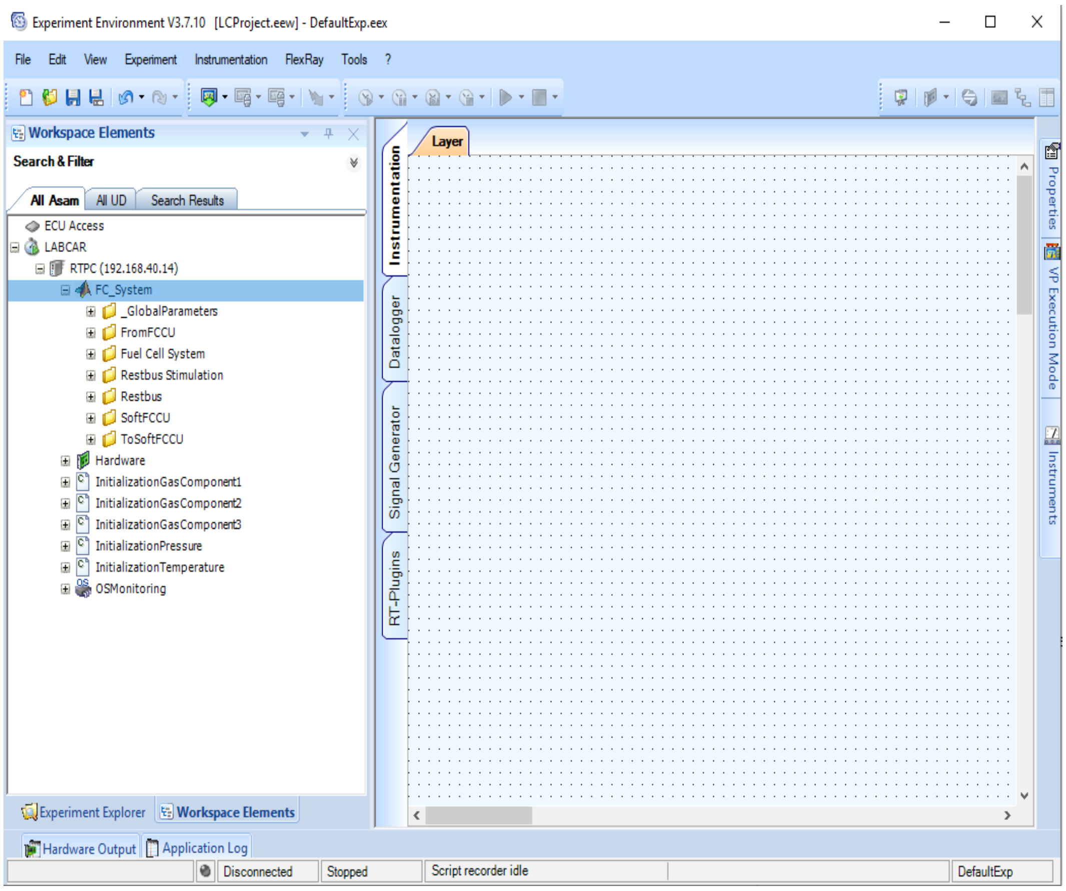
To build a LABCAR-OPERATOR project

1. Choose **Project** → **Build**.
2. Activate the code generation for the "FC_System" module.
3. Click **Build**.

For more details about Simulink Code Generation, refer to the section “Simulink Model - Code Generation Process” on page 14.

7.1.3 Experiment Creation

After the successful build of the project in COSYM or LABCAR-OPERATOR, you can prepare the experiment for the first execution of the simulation model. Starting point is an empty experiment.

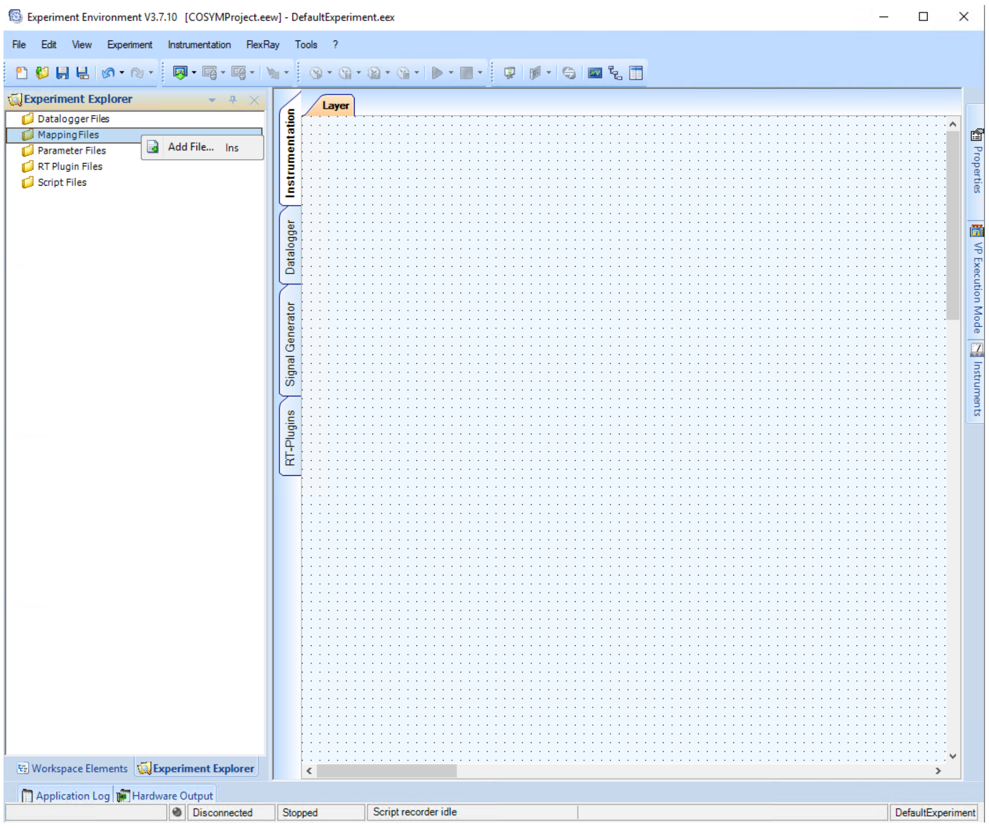


Mapping Files

"Mapping files" provide a possibility to simplify the signal handling in LABCAR-EE. For LABCAR-MODEL-FC various signal mappings are pre-defined. This allows a simple access to most important simulation signals without in-depth knowledge of the model.

1. Open **Experiment Explorer.**
2. Add all mapping files located in the install folder at

   `Project Items\LCO\FC_Default_StackSoftFCCU\mapping`



NOTE

Ensure that the added mapping files are also activated.

Parameter Files

No parameter files are provided. The default values of the Simulink model are used.

Layers

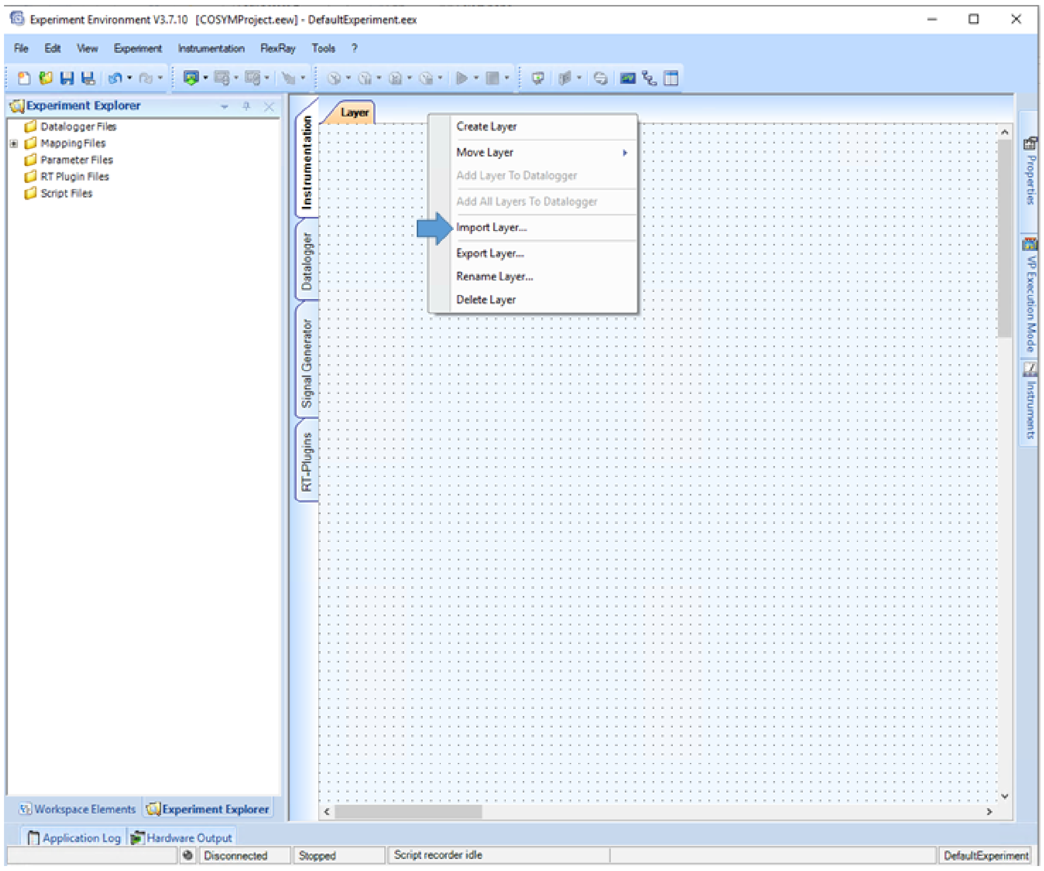
The "Layer" provides the basic interface to handle the vehicle model. It allows the setting of the driver mode and the selection of pre-defined cycles. As a preliminary step, you can copy images for the layer from the installation folder.

To import layers

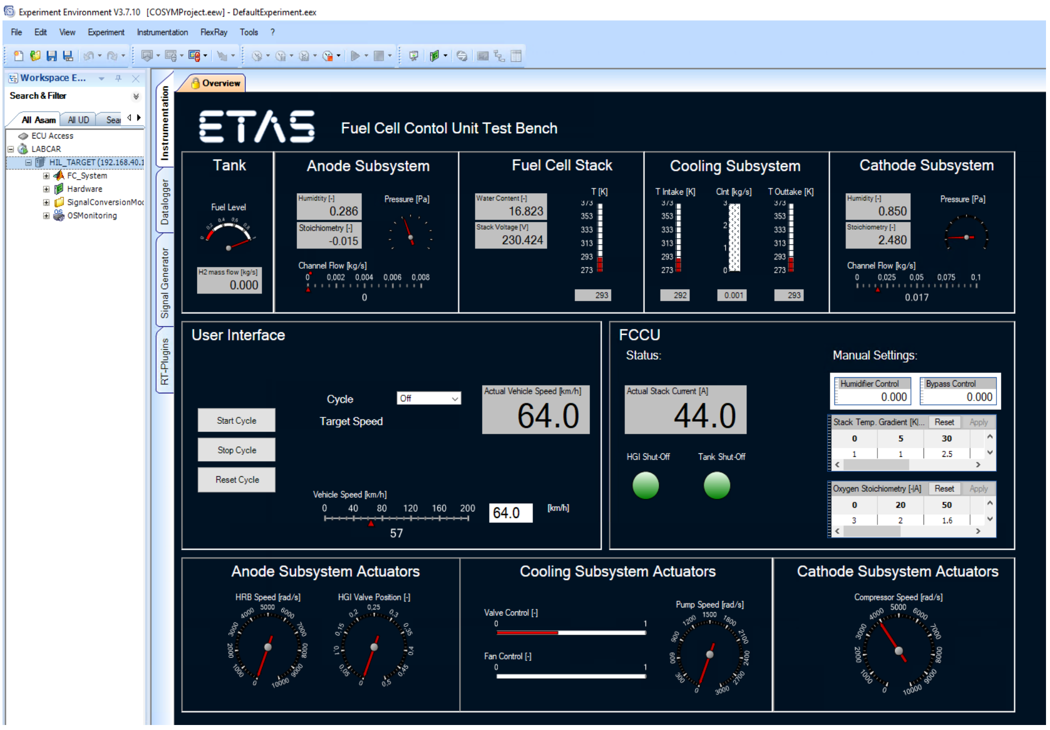
1. Copy images located in install folder at

   Project Items\LCO\FC_Default_StackSoftFCCU\img.
2. Create the target folder if it does not already exist in the Experimentation folder and name it as "Image_Files".
3. Paste the images copied in step 1.

4. Import the `FuelCell_Overview.layer` file as located in `Project Items\LCO\FC_Default_StackSoftFCCU\gui`.



After the successful import of the `FuelCell_Overview.layer`, basic dashboards for handling the fuel-cell system model exist.


Experiment Environment V3.7.10 [COSYMProject.eew] - DefaultExperiment.eex
File Edit View Experiment Instrumentation FlexRay Tools ?
Overview
ETAS Fuel Cell Contol Unit Test Bench
Tank
Anode Subsystem
Fuel Cell Stack
Cooling Subsystem
Cathode Subsystem
User Interface
FCCU
Anode Subsystem Actuators
Cooling Subsystem Actuators
Cathode Subsystem Actuators
Start Cycle
Stop Cycle
Reset Cycle

> **NOTE**
>
> All pre-defined instruments use mapped signals provided by the mapping files. For proper execution, add all mapping files to the experiment.

> **NOTE**
>
> Consider that only default LABCAR-EE instruments are used. You can adapt the layer individually to your needs.

All required artifacts for the first execution of the models are added to the experiment. With that, the experiment is ready for the download to RTPC and the simulation. The section "Getting Started with the Tutorial Project" on page 151 describes the handling of the simulation model in more detail.

7.2 Getting Started with the Tutorial Project

The following tutorial gives an introduction on how to use LABCAR-MODEL-FC. Within this tutorial LABCAR-MODEL-FC is used to represent an automotive fuel-cell system. A physical control unit is not required, since the SoftFCCU of LABCAR-MODEL-FC will be used to control the model.

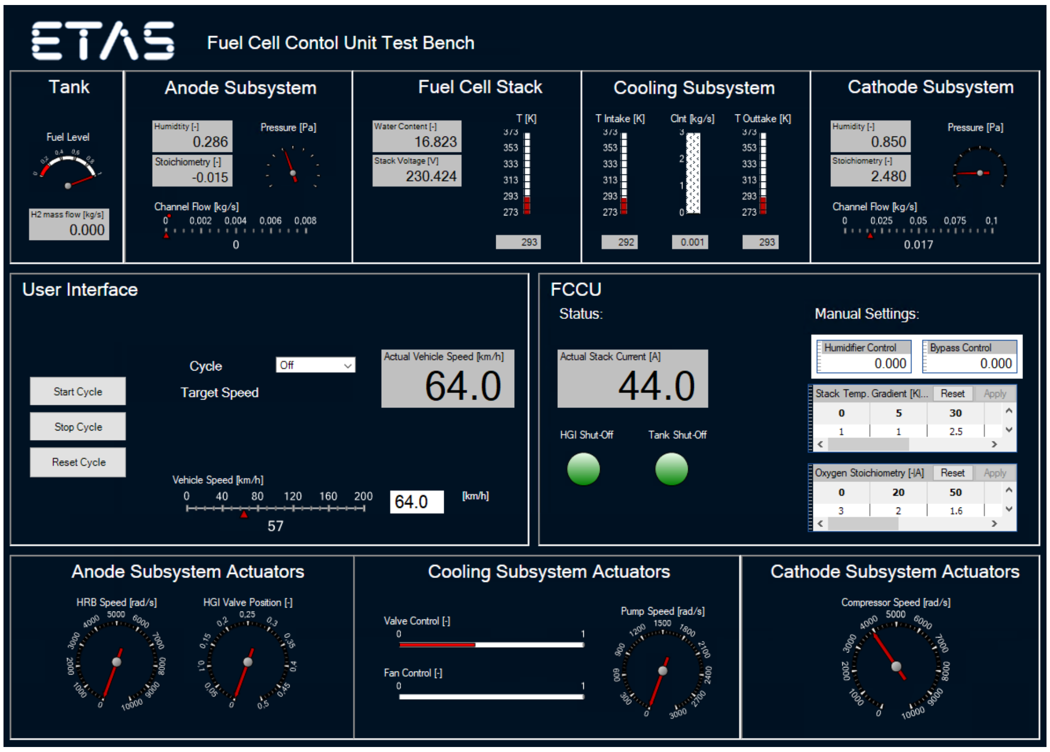
The tutorial project is located in `.\Example Project`.

NOTE

Please do not open or manipulate the example project in the installation folder directly. Instead, copy the example project from the installation folder to a location of your choice.

7.2.1 Overview

The provided graphical user interface for LABCAR-MODEL-FC contains basic functionality to operate the vehicle model comfortably. The GUI allows to select different operation modes to stimulate the fuel-cell system model. Basic observables about the operation of fuel-cell system, actuators, and SoftFCCU are presented graphically.



- The upper row in the overview represents the most significant measurement quantities within the fuel-cell system. They include information about the stack and corresponding supply systems, namely hydrogen, air and the coolant supply system.
- The "User Interface" section provides controls to stimulate the fuel-cell system. After selecting the cycle mode, you can stimulate the fuel-cell system by interacting with the available controls. The section “User Interface” on page 157 describes the individual control elements.
- The section "FCCU" shows the requested current and the status of the Shut-Off valves. Moreover, most important SoftFCCU parameters are given, to adapt the applied operation strategy of the FCCU.

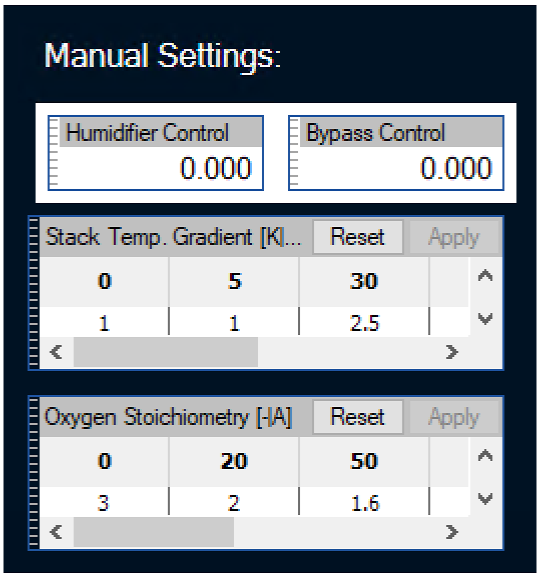
- The lower row in the overview represents the status of the actuators in the supply systems. In detail, these are the rotational speeds of the pumps, blower, and compressor as well as valve positions.

NOTE

The GUI uses standard instruments of LABCAR-EE and an open-source RT-Plugin to extend functionality of the instruments. This can easily be used as a template for individual GUIs.

7.2.2 Manual Control of Humidifier and Bypass

Humidifier and bypass are closed by default. They are controlled manually by using the manual settings in the section "FCCU" of the GUI. You can perform according adjustments via the inputs "Humidifier Control" and "Bypass Control" of the SoftFCCU. Both parameters are normalized actuator inputs. Hence, valid values are between zero and one. By that, cathode bypass and humidifier can be activated or deactivated.



Manual Settings:

Humidifier Control 0.000

Bypass Control 0.000

Stack Temp. Gradient [K]...

0	5	30
1	1	2.5

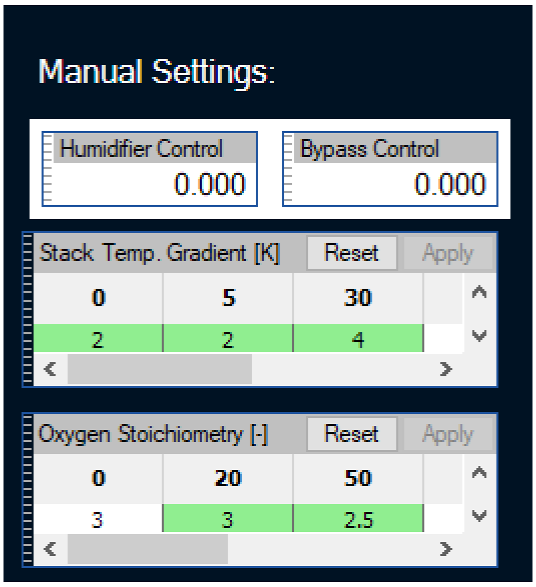
Oxygen Stoichiometry [-|A]

0	20	50
3	2	1.6

Fig. 7-4 Manual Setting Section to Activate or Deactivate Cathode Humidifier and Bypass

7.2.3 Adaptation of the Operation Strategy

The stack temperature gradient and the oxygen stoichiometry are key entities to characterize the operation strategy applied to a system. They influence the water management and the efficiency of the electro-chemical reaction inside the fuel cells. The SoftFCCU has a default strategy implemented, which you can revise and change. For that, the "FCCU" section of the GUI contains inputs for the "Stack Temp. Gradient" and "Oxygen Stoichiometry". Both values depend on the current operation state. Hence, instead of scalars, one-dimensional maps exist, which depend on the requested stack current.



Manual Settings:

Humidifier Control 0.000

Bypass Control 0.000

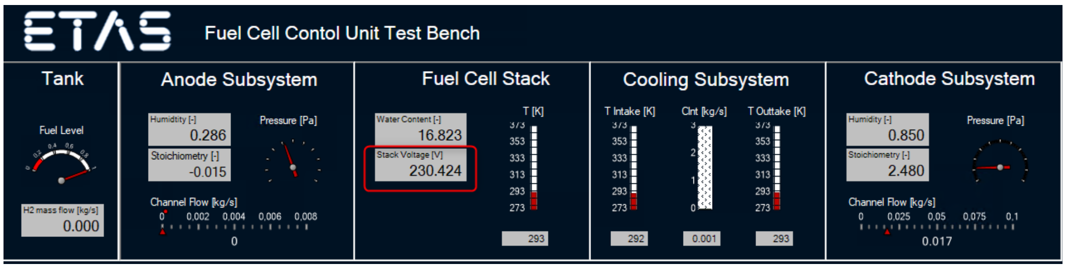
Stack Temp. Gradient [K]		
0	**5**	**30**
2	2	4

Oxygen Stoichiometry [-]		
0	**20**	**50**
3	3	2.5

Fig. 7-5 Manually Adapted Setting for the Operation Strategy of the SoftFCCU

After you adjusted the values, press "Apply" to activate your changes. With that, the SoftFCCU calculates new desired values for the operation of the stack, e.g. by adapting the oxygen or coolant mass flow. Based on the physical representation of the stack, the expected influence on the stack can be revised.

Example: You can increase the desired oxygen stoichiometry using the instrument. When applying the changes to the SoftFCCU, the rotational speed of the air compressor should increase accordingly. More air mass flow is provided to the stack. A positive influence on the stack efficiency can be expected. The stack voltage raises.



Fuel Cell Contol Unit Test Bench

Tank: H2 mass flow [kg/s] 0.000

Anode Subsystem: Humidtity [-] 0.286; Stoichiometry [-] -0.015; Channel Flow [kg/s] 0

Fuel Cell Stack: Water Content [-] 16.823; Stack Voltage [V] 230.424; T [K] 293

Cooling Subsystem: T Intake [K] 292; Clnt [kg/s] 0.001; T Outtake [K] 293

Cathode Subsystem: Humidity [-] 0.850; Stoichiometry [-] 2.480; Channel Flow [kg/s] 0.017

Fig. 7-6 Increasing/Decreasing Stack Voltage

The effect of a changed operation strategy can be revised by the reaction of the fuel-cell stack, e.g. an increasing or decreasing stack voltage.

7.3 Fuel-Cell Subsystems and Components

The current version of LABCAR-MODEL-FC includes a single reference system configuration. It demonstrates how to use the individual components. Different connections of the components are possible, so that other FC system topologies are feasible as well.

7.3.1 Cathode Supply System

The cathode supply system feeds oxygen to the stack. It consists out of a single stage compressor driven by an e-machine. It contains an intercooler, humidifier, and a stack bypass for further control interactions. The model architecture is given in Fig. 7-7.

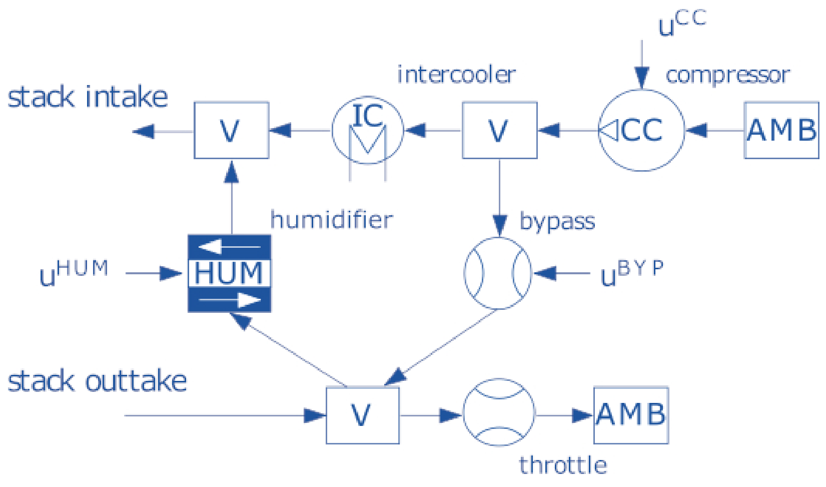


Fig. 7-7 Cathode Supply System Components View

Please refer to the Model Reference document section 2 for a detailed description of the cathode supply system and its components including a description of the complete interface.

7.3.2 Hydrogen Supply System

The hydrogen/anode supply system feeds hydrogen to the stack. It consists of a supply path and a recirculation loop. The supply path contains a hydrogen tank, pressure reducer, and a hydrogen gas injector. The recirculation loop contains a water separator as well as a recirculation blower. An additional purge valve is available and rejects inert gases to the ambient.

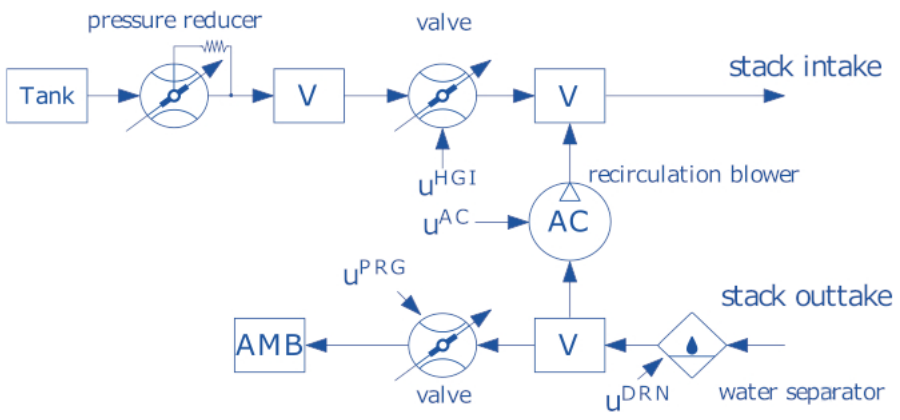


Fig. 7-8 Hydrogen Supply System Components View

Please refer to the Model Reference document section 4 for a detailed description of the hydrogen supply system and its components including a description of the complete interface.

7.3.3 Coolant Supply System

Fuel-cell stacks generate a notable amount of waste heat at increasing current densities. Nevertheless, the low stack temperature limits the heat rejection via exhaust gas and heat exchange with the ambient temperature. A separate cooling subsystem is required and the provision of sufficient cooling power is a central point in the functional development as well as the system design.

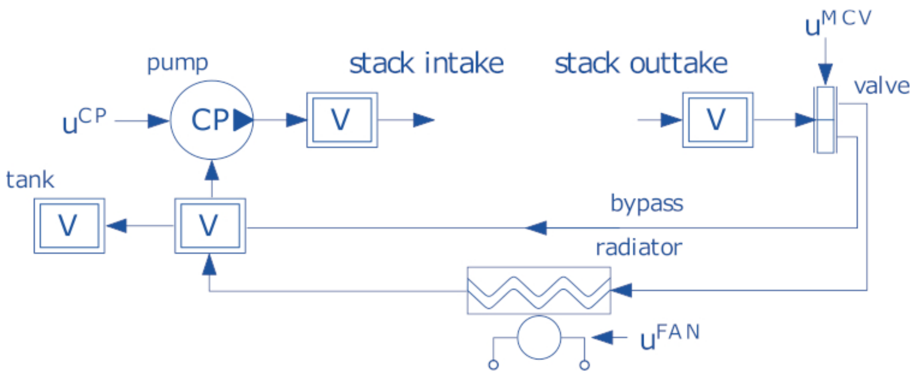


Fig. 7-9 Coolant Supply System Components View

Please refer to the Model Reference document section 3 for a detailed description of the coolant supply system and its components including a description of the complete interface.

7.3.4 Stack

LABCAR-MODEL-FC represents a 1-D+1-D fuel-cell stack model. The model includes a 1-D membrane model for a detailed description of the membrane resistance, water content, and the resulting water exchange between the electrodes. Furthermore, a spatial distributed 1-D bipolar plate and gas channel model is used to consider the non-linear characteristics, e.g. temperature, current, and pressure variations along the stack. Especially automotive relevant phenomena, e.g. influence of subfreezing temperatures, are considered.

The following features are regarded:

- Single cell model with a detailed loss behavior considering the affect of current, temperature, reactant stoichiometry, and membrane humidity on cell voltage
- Detailed calculation of water content and water exchange based on a 1-D membrane model
- 1-D multi-component gas channel model allows the specification of individual gas composition for each electrode

  Simple to define pressure loss characteristics
- Simulation of different flow field designs

  Co/Counter flow setups with support of internal cell humidification
- Realistic cold start-up behavior based on membrane's temperature model, non-linear dynamics of the cells water content, and temperature dependent fluid properties
- Two-phase water model considering accumulation and movement of liquid water in the gas channel

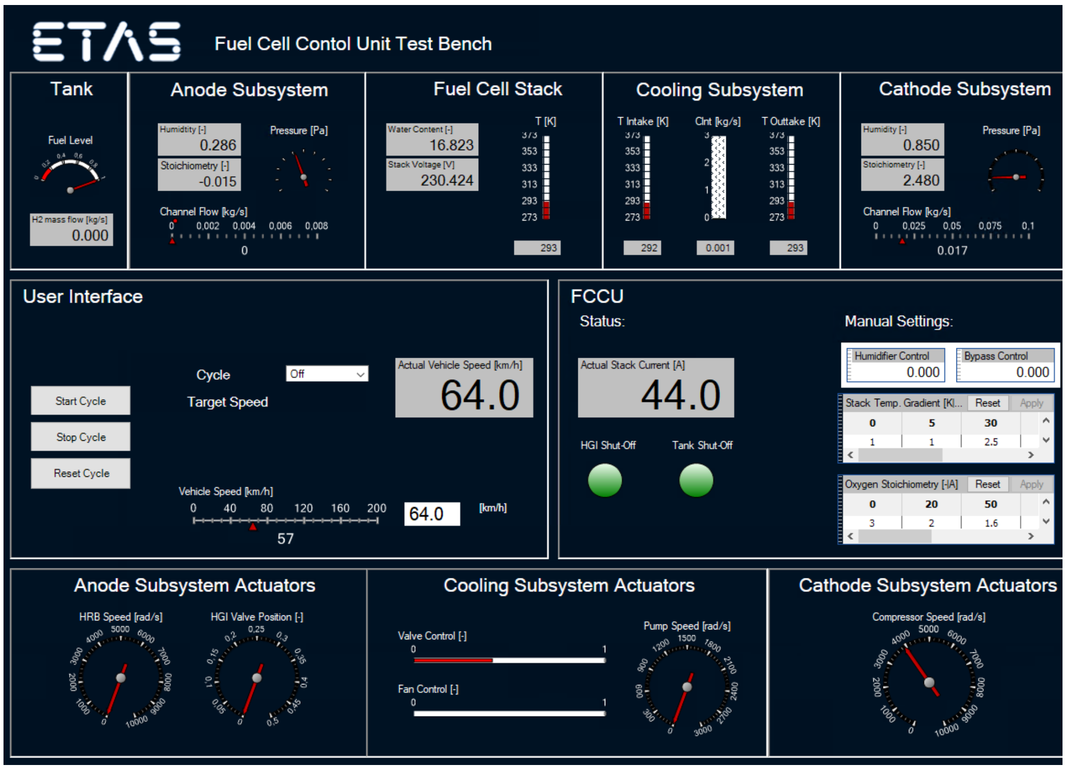
Please refer to the Model Reference document section 5 for a detailed description of the fuel-cell stack and its components including a description of the complete interface.

7.4 User Interface

The provided graphical user interface for LABCAR-MODEL-FC contains basic functionality to comfortably operate the fuel-cell model. The GUI enables you to either set a desired vehicle speed manually, use external stimulation or use a pre-defined cycle. The fuel-cell model is controlled by a basic fuel-cell control unit, which will get the demanded current as a request. Furthermore, the GUI provides a native instrument to observe the current condition of the fuel-cell model.

7.4.1 Overview

Operating and observing is the main focus, such that the GUI represents the instrumentation for user interaction and measuring instrumentation to observe several effects. Therefore, the GUI is arranged in several sections as the following image demonstrates.



The upper row in the overview represents the most significant measurement quantities within the fuel cell between corresponding supply systems and the stack.

- The sections "Tank", "Anode Subsystem", "Fuel Cell Stack", "Cooling Subsystem", and the "Cathode Subsystem" show the current condition of the fuel-cell model by demonstrating temperatures, pressures, stoichiometry, humidity, and mass flows.
- With the section "User Interface" you are able to start the fuel-cell system, to manually set a desired vehicle speed, or to choose a drive cycle. The vehicle speed is mapped\translated to the current request based on a simple mapping table. There is no dedicated vehicle model.
- The section "FCCU" shows an overall identification of the system.
- With the bottom sections "Anode Subsystem Actuators", "Cooling Subsystem actuators", and "Cathode Subsystem actuators" you are able to observe the operation behavior of the actuators while operated by the SoftFCCU.

7.4.2 GUI Section User Interface

The section "User Interface" allows to operate the fuel-cell model.

You can choose from the drop-down list several operation modes. The predefined modes are:

- Manual:

  You can set the vehicle speed either by moving the slider or by directly entering the vehicle speed in the input field.
- External:

  You can set a user-defined vehicle speed trajectory which will be followed.
- Cycle:

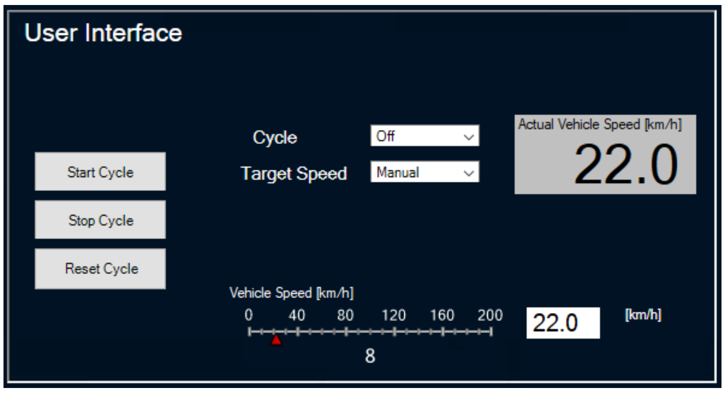
  You can choose from pre-defined driving cycles. If the mode "Cycle" is selected, the "Cycle" drop-down list applies. Here you can choose a driving cycle of interest.

  Pre-defined cycles are:

  – WLTP

  – User defined

  – Off



7.4.3 GUI Sections for Observation

The upper sections of the GUI provide the basic information about the anode supply system, the cathode supply system, the cooling supply system, and the fuel-cell stack.

- Tank

  The tank is part of the anode supply system and indicates the current hydrogen level; additionally it displays how much hydrogen is flowing to the anode.
- Anode Subsystem

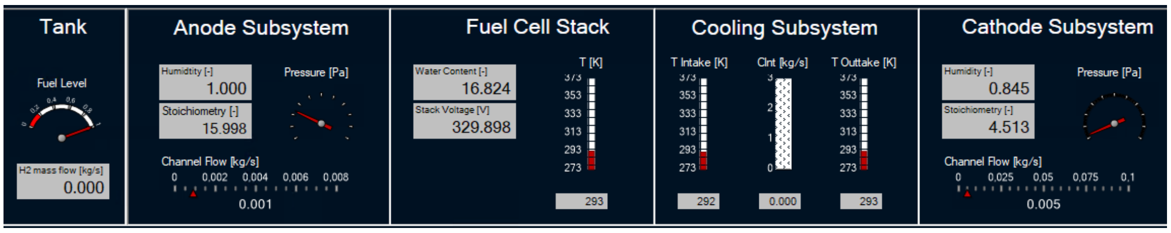
  This section provides the information about the anode side of the fuel cell. Typically, a pressure sensor observes the pressure at the intake manifold. Furthermore, it shows the humidity, the stoichiometry at the anode, and how much hydrogen is provided to the anode.
- Fuel Cell Stack

  This section provides general stack information. You can observe the current water content within the stack and also the voltage level at the electrical stack terminal. As an indication in which operation range the fuel cell is operating, the current efficiency is demonstrated.

- Cooling Subsystem

  During operation, the generated heat within the stack has to be transported. You can observe the current temperatures at the intake manifold and the outtake manifold of the cooling supply system.
- Cathode Subsystem

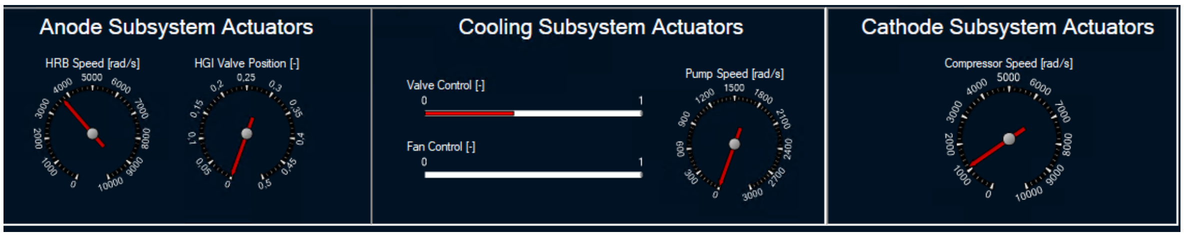
  This section provides the information about the cathode side of the fuel cell. Typically, a pressure and a temperature sensor observe the conditions at the intake manifold. Furthermore, it shows the humidity, the stoichiometry at the cathode, and how much oxygen is provided to the cathode.



The lower sections of the GUI provide basic information about the actuators of the corresponding supply system. The GUI contains illustrations for the main components. These components are:

- Hydrogen Recirculation Blower (Anode Supply System)
- Hydrogen Injector Valve (Anode Supply System)
- Compressor (Cathode Supply System)
- Mass Control Valve (Cooling Supply System)
- Radiator (Cooling Supply System)
- Pump (Cooling Supply System)

You can observe several actuators. The valves are observed by demonstrating their current position. The compressor, the hydrogen recirculation blower, and the pump are observed by demonstrating their shaft speed when actuated by the SoftFCCU. The following picture shows this specific section.



8 Licensing

The simulation models of the LABCAR-MODEL product family possess a license protection. Each product of the LABCAR-MODEL family requires its own license and refers to a specific version. The actual mechanism differs between FPGA-based and Simulink-based models; they are described in the following sections.

8.1 FPGA-Based Simulation Models

The LABCAR-MODEL-PMSM runtime license is generated based on the serial number of the ES5340.2M board. The serial number can be obtained from the "HSP" tool using the "Hardware" window. A Runtime license file consists of values (Hexadecimal) for the model license inputs "Lic_0", "Lic_1", "Lic_2", "Lic_3" and "Lic_4". A LABCAR-EE layer "PMSM_Runtime_License" is available to enter runtime license keys. Nevertheless the following mechanism can be used to apply license values.

- Browse "RTPC/Hardware/_.../Es5340flexiblefpga" in LABCAR-EE to access the model license inputs.
- Use the "Signal-Calibrate Constant Value"option to edit input values.
- Change the data representation from decimal to hexadecimal for "Edit Box" of every model license input.
- Provide the license values given in the runtime license file for "Lic_0" to "Lic_4".

8.1.1 License Status Information

You can retrieve the licensing status in the LABCAR-EE from model license output "Lic_out". "Lic_out" is "1" or "License Status" is green in "PMSM_Runtime_License" layer if correct license values are set for model license inputs. Otherwise it is "0" or the "License Status" is Grey.

8.2 Simulink-Based Simulation Models

Two types of licenses exist for the Simulink-based simulation models. They are:

- A *Runtime License* ("RT-License") permits the execution of a model containing one or more S-Function blocks from LABCAR- MODEL, irrespective of the underlying platform (e.g., Windows or ETAS RTPC).
- An *Operator License* permits the code-generation of a model containing one or more S-Function blocks from LABCAR-MODEL. In addition, it permits the execution of a model.

On Windows machines, the ETAS License Manager is used to verify the LABCAR-MODEL license.

You find the procedure to install license files on ETAS RTPC in the LABCAR-OPERATOR documentation [5].

8.2.1 License Status Information (Windows)

The license status is given in the ETAS License Manager. The MATLAB command window also reports the licensing status. An error occurs, if the ETAS License Manager is not able to find a valid license or when the license checkout fails. In this case, the error message is written to the MATLAB command window. The error message includes details about the S-Function, which caused the error.

8.2.2 License Status Information (RTPC)

When using ETAS RTPC as simulation target, the license status is printed to the hardware output of LABCAR-EE during the model initialization. The level of detail of the printed messages is determined by the logging level of the RTPC. Please see [5] for further information.

Glossary

This section explains technical terms used in this manual. The abbreviations and terms are listed in alphabetic order.

Abbreviations

BMS

Battery Management System, i.e., the ECU that controls an automotive battery.

ECU

Electronic Control Unit

EMF

Electromotive Force

EMS

Engine Management System, i.e., the ECU that controls the combustion engine

FPGA

Field Programmable Gate Array

FC

Fuel Cell

FCCU

Fuel Cell Control Unit

FMU

Function Mock-up Unit

GUI

Graphical User Interface

HiL

Hardware-in-the-Loop

HSP

Hardware Service Pack

ICE

Internal Combustion Engine

IGBT

Insulated Gate Bipolar Transistor

LCRT

LABCAR-RUNTIME

MCU

Motor Control Unit, i.e., an ECU that controls an electrical motor

MiL

Model-in-the-Loop

OSS

Open Source Scan

PEM

Polymer Electrolyte Membrane

PMSM

Permanent Magnet Synchronous Motor; an electrical motor

PWM

Pulse Width Modulation

SiL

Software-in-the-Loop

SUT

System Under Test

VVTB

Virtual Vehicle Test Bench

XiL

X-in-the-Loop; X is a placeholder for H, S, or M. XiL is the superset of MiL, SiL, and HiL solutions.

Terms

Offline Parametrization

Executing the simulation in LABCAR-EE. The unit under test, either a physical or a virtual ECU, is disconnected (hence "offline"). The set with the parametrization is stored in a supported parameter file.

Online Parametrization

The unit under test, either a physical or a virtual ECU, is connected (hence "online"). The project is linked to the parameter file from the offline parametrization. The simulation is running.

References

For more information refer to the following documents that are part of the documentation package of LABCAR-OPERATOR.

[1] LABCAR-OPERATOR V5.4.10 User´s Guide
[2] LABCAR-OPERATOR V5.4.10 Software Compatibility List
[3] ES5340.2 Electric Drive Simulation Board User's Guide
[4] FlexibleFPGA V3.0.0 - Workflow Description: Tutorial
[5] ETAS RTPC User´s Guide

For more information refer to the following documents that are part of the documentation package of COSYM.

[6] COSYM V2.3.0 User's Guide
[7] COSYM_V2.3.0_Software_Compatibility_List

Figures

Contact Information

ETAS Headquarters

ETAS GmbH

Borsigstraße 24	Phone:	+49 711 3423-0
70469 Stuttgart	Fax:	+49 711 3423-2106
Germany	Internet:	www.etas.com

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