# Contents

1  | Introduction .................................................................................. 6  
1.1 | About this manual ....................................................................... 6  
1.2 | Safety Information ...................................................................... 6  
1.3 | Privacy ......................................................................................... 7  
1.3.1 | Data Processing ......................................................................... 7  
1.3.2 | Data and Data Categories .............................................................. 7  
1.3.3 | Technical and Organizational Measures ........................................ 7  
1.4 | Conventions .................................................................................. 7  
1.4.1 | Documentation Conventions .......................................................... 7  
1.4.2 | Typographic Conventions ............................................................... 8  
2  | LABCAR-MODEL Family .................................................................. 9  
2.1 | Overview of Models ...................................................................... 9  
2.2 | General Model Parametrization ....................................................... 9  
2.2.1 | Obtaining the Data for Parametrization .......................................... 9  
2.2.2 | Model Parametrization Process .................................................... 10  
2.2.2.1 | General Parametrization Process ............................................... 10  
2.2.2.2 | Offline Parametrization ............................................................... 11  
2.2.2.3 | Online Parametrization ............................................................... 11  
2.3 | Technical Aspects ......................................................................... 13  
2.3.1 | LABCAR Port Blocks .................................................................. 13  
2.3.2 | Simulink Model - Code Generation Process .................................... 14  
2.3.3 | MEX File Version Information ..................................................... 15  
3  | Installation .................................................................................... 16  
3.1 | User Privileges Required for Installation ......................................... 16  
3.2 | Contents of the DVD ..................................................................... 16  
3.3 | Program Installation .................................................................... 16  
3.4 | Contents of the Installation ............................................................ 17  
4  | LABCAR-MODEL-VVTB .................................................................. 19  
4.1 | Project and Experiment Creation ................................................... 19  
4.1.1 | Project Creation in LABCAR-OPERATOR ..................................... 19  
4.1.2 | Experiment Creation with LABCAR-OPERATOR .......................... 23  
4.2 | Getting Started with the Tutorial Project ....................................... 28  
4.2.1 | Overview .................................................................................... 29  
4.2.2 | Cycle Driving .............................................................................. 30  
4.2.2.1 | Predefined Cycles ....................................................................... 30  
4.2.2.2 | User-Defined Cycle .................................................................... 31  
4.3 | Signal Interface ........................................................................... 35  
4.3.1 | Driver ......................................................................................... 35  
4.3.2 | SoftECU ....................................................................................... 35  
4.3.3 | Vehicle/Drivetrain ....................................................................... 36  
4.3.4 | Vehicle/CombustionEngine .......................................................... 37  
4.3.5 | Driver/BodyComponents .............................................................. 37  
4.4 | User Interface ............................................................................... 38
## 5 LABCAR-MODEL-ICE

- **5.1 Project and Experiment Creation**
  - 5.1.1 Project Creation in LABCAR-OPERATOR
  - 5.1.2 Experiment Creation with LABCAR-OPERATOR

- **5.2 Getting Started with the Tutorial Project**
  - 5.2.1 Overview
  - 5.2.2 Model Control Modes
  - 5.2.3 Manipulation of Air Mass Flow into Cylinder Through Throttle Plate

- **5.3 Signal Interface**
  - 5.3.1 Combustion Engine/Top Level
  - 5.3.2 Combustion Engine/Injection
  - 5.3.3 Combustion Engine/Lambda Sensor
  - 5.3.4 Combustion Engine/Throttle Plate
  - 5.3.5 Combustion Engine/Engine Torque

- **5.3.5 Combustion Engine/Engine Torque**
  - Combustion Engine/Engine Torque

## 6 LABCAR-MODEL-PMSM

- **6.1 Overview**
- **6.2 Project and Experiment Creation**
  - 6.2.1 Flashing LABCAR-MODEL-PMSM to ES5340.2-M
  - 6.2.2 Project Creation in LABCAR-OPERATOR
  - 6.2.3 Experiment Creation with LABCAR-OPERATOR
- **6.3 Getting Started with Tutorial Project**
  - 6.3.1 Overview
  - 6.3.2 Active Short-Circuit Operation Mode
- **6.4 Nonlinear Maps and Parameter Maps Downloader**
- **6.5 Signal Interface**
  - 6.5.1 Drivetrain Interfaces
  - 6.5.2 Parameters
  - 6.5.3 Inverter Interfaces
  - 6.5.4 PMSM Motor Interfaces
  - 6.5.5 DAC Interfaces
  - 6.5.6 Digital Interfaces
- **6.6 User Interfaces**
  - 6.6.1 PMSM Runtime License
  - 6.6.2 PMSM Simulation Control
  - 6.6.3 PMSM DAC
  - 6.6.4 PMSM Failure Simulations
  - 6.6.5 PMSM Parameters

## 7 Licensing

- **7.1 FPGA-Based Simulation Models**
  - 7.1.1 License Status Information
- **7.2 Simulink-Based Simulation Models**
  - 7.2.1 License Status Information (Windows)
  - 7.2.2 License Status Information (RTPC)

- **Glossary**
# Contents

<table>
<thead>
<tr>
<th>References</th>
<th>101</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figures</td>
<td>102</td>
</tr>
<tr>
<td>ETAS Contact Addresses</td>
<td>103</td>
</tr>
</tbody>
</table>
1 Introduction

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

1.1 About this manual

This manual describes the handling and the licensing of the LABCAR-MODEL simulation models. This manual contains the following chapters:

- "LABCAR-MODEL Family" on page 9
- "Installation" on page 16
- "LABCAR-MODEL-VVTB" on page 19
- "LABCAR-MODEL-ICE" on page 46
- "LABCAR-MODEL-PMSM" on page 64
- "Licensing" on page 97
- "Abbreviations" on page 99 and "Terms" on page 100
- "References" on page 101

1.2 Safety Information

The safety messages contained in this manual are identified with the danger symbol shown below:

⚠️

The safety messages shown below are used for this purpose. They provide notes to extremely important information. Please read this information carefully.

⚠️ **DANGER**

Indicates an immediate danger with a high risk of death or serious physical injury if not avoided.

⚠️ **WARNING**

Indicates a possible danger with moderate risk of death or (serious) physical injury if not avoided.

⚠️ **CAUTION**

Identifies a hazard with low risk of minor or moderate physical injury.
1.3 Privacy

Your privacy is important to ETAS. We have created the following privacy statement 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.

1.3.1 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.

1.3.2 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

1.3.3 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.

1.4 Conventions

1.4.1 Documentation Conventions

All actions to be performed by the user are presented in a so-called “use-case” format. This means that the objective to be reached is first briefly defined in the title, and the steps required to reach the objective are then provided in a list.

This presentation appears as follows:

**Definition of Objective**

Any preliminary information...

1. Execute action 1.

Any explanation for action 1...
2. Execute action 2.
Any explanation for action 2...
3. Execute action 3.
Any explanation for action 3

Any concluding remarks...
A specific example:

**To create a new file**
When creating a new file, no other file may be open.

1. Choose **File → New**.
The "Create file" dialog box appears.
2. Type the name of the new file in the "File name" field.
The file name must not exceed 8 characters.
3. Click **OK**.
-> The new file will be created and saved under the name you specified. You can now work with the file.

### 1.4.2 Typographic Conventions
The following typographic conventions are used:

OCI_CANTxMessage msg0 = Code snippets are presented in the **Courier** font.
Meaning and usage of each command are explained by means of comments.
The comments are enclosed by the usual syntax for comments.

Select **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 block capitals.
The 'Open File' dialog box appears.

Select the file **setup.exe**. Names of program windows, dialog boxes, fields etc. are shown in quotation marks.

A conversion between the file types logical and arithmetic is **not** possible.

See [1] References to further literature are shown in **italics**.

**NOTE**
Important hint for the user.
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-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 chapters.

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.
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.

![Online parametrization workflow](image-url)
2.3 Technical Aspects

This chapter 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](image)

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 LABCAR-EE by overwriting the current signal value.

<table>
<thead>
<tr>
<th>Access</th>
<th>Simulink Inport</th>
<th>Simulink Outport</th>
<th>LCInport</th>
<th>LCOutport</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Logging</th>
<th>Simulink Inport</th>
<th>Simulink Outport</th>
<th>LCInport</th>
<th>LCOutport</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Manipulation</th>
<th>Simulink Inport</th>
<th>Simulink Outport</th>
<th>LCInport</th>
<th>LCOutport</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stimulation</th>
<th>Simulink Inport</th>
<th>Simulink Outport</th>
<th>LCInport</th>
<th>LCOutport</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

**Tab. 2-1 SIMULINK port and LCPort functionality in LABCAR-OPERATOR**
2.3.2 **SIMULINK Model - Code Generation Process**

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 LABCAR 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 that 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 code generation with the lcrt.tlc file that is shipped with LABCAR-OPERATOR.

![Code generation process for SIMULINK models](image)

**NOTE**

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

This chapter 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.

![Image of setup process]
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 pre-build SIMULINK models including compiled S-Fcns in the form of mexw64 files for windows and o-files for the RTPC execution.

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

- **Project Items:**
  - These are additional LABCAR-OPERATOR compatible items provided as a bundle. The items comprise:
    - C-modules (cmd)
    - connection files (conn)
    - layer (gui)
    - images (img)
    - mapping files (mapping)
    - rt-plugins (rtp)
    - and specific user-data (userdata).

  The bundles are specific for particular model variants, e.g. connection files between automatic transmission model (VVBT_AT) and manual transmission model (VVTB_MT) differ and are individually stored.
• LicenseConfigurationInfo:
  These are files for the ETAS License Manager.
• Example Project:
  Here you find pre-build LABCAR-OPERATOR example projects.
• 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, a step-by-step guide is given, which describes the generation of 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 LABCAR-OPERATOR

This chapter 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 chapter.

To set up the LABCAR-OPERATOR project, do the following:

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 out 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

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\_Lib\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.
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\Lib\VVTB_MT\conn. 
   -> After the connection files are added, many connections are visible in the "Connection Manager".

**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 checkbox for the VVTB module.
3. Click **Build**.

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.
4.1.2 Experiment Creation with LABCAR-OPERATOR

After the successful build of the project, 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-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\Lib\VVTB_MT\mapping.
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\_Lib\VVTB_MT\rtp\Driver _Instruments.

NOTE

Ensure that the added mapping files are also activated.
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.
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 
   \VV\VVTB\Project Items\_Lib\VVTB_MT\img.
2. Create the target folder if it does not already exist.
3. Import the Driver.layer file as located in 
   \VV\VVTB\Project Items\_Lib\VVTB_MT\gui.

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

**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 21.
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 chapter “Getting Started with the Tutorial Project” on page 28 describes the handling of the simulation model in more detail.

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.
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.

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.
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** → **Target Speed - External Gear (MT)**

In the given example, the driver determines the gear.

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

  ![Diagrams](image)

  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.
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)**

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

  Set **Cycle Mode** → **Usr**

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

- Search cycle
- Drag & Drop D_TestCycles__UserdefinedCurve
- Edit D_TestCycles__UserdefinedCurve
The user-defined cycle is defined by the parameter $D_{TestCycles\_UserdefinedCurve}$, which can be adapted accordingly. Start the simulation and 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:

  \[ \text{Driver Mode} \rightarrow \text{Target Speed - Internal Gear (MT)} \]

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

\[ \text{Target Speed} \rightarrow \text{External} \]

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 LCImport, set following options:

- "Mode" → SIGNALGENERATOR
- "Signal Description" → SignalGenerator/…

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.
4.3 Signal Interface

All of the presented interfaces connect driver, SoftECU, and vehicle in the example project of the chapter “Project and Experiment Creation” on page 19. 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*

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b_Ignition</td>
<td>Bool</td>
<td>-</td>
<td>Driver ignition request</td>
</tr>
<tr>
<td>b_Starter</td>
<td>Bool</td>
<td>-</td>
<td>Driver starter request</td>
</tr>
<tr>
<td>i_Gear</td>
<td>Int</td>
<td>-</td>
<td>Desired gear</td>
</tr>
<tr>
<td>r_Accelerator_0_1</td>
<td>Double</td>
<td>-</td>
<td>Accelerator pedal position</td>
</tr>
<tr>
<td>r_Brake_0_1</td>
<td>Double</td>
<td>-</td>
<td>Brake pedal position</td>
</tr>
<tr>
<td>r_Clutch_0_1</td>
<td>Double</td>
<td>-</td>
<td>Clutch pedal position</td>
</tr>
</tbody>
</table>

*Outputs*

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b_Ignition</td>
<td>Bool</td>
<td>-</td>
<td>Driver ignition signal</td>
</tr>
<tr>
<td>n_Engine</td>
<td>Double</td>
<td>rpm</td>
<td>Measured engine speed</td>
</tr>
<tr>
<td>M_Ind_In</td>
<td>Double</td>
<td>Nm</td>
<td>Measured induced engine torque</td>
</tr>
<tr>
<td>M_Drag_In</td>
<td>Double</td>
<td>Nm</td>
<td>Measured engine drag torque</td>
</tr>
<tr>
<td>M_CombEngine</td>
<td>Double</td>
<td>Nm</td>
<td>Effective engine torque</td>
</tr>
<tr>
<td>N_ThrottlePlateOpen-ing_r_0_100</td>
<td>Double</td>
<td>%</td>
<td>Measured throttle plate position</td>
</tr>
<tr>
<td>In_Gearbox_Shifting</td>
<td>Bool</td>
<td>-</td>
<td>Flag indicating active gearbox shifting</td>
</tr>
<tr>
<td>Ignition</td>
<td>Bool</td>
<td>-</td>
<td>Driver ignition flag</td>
</tr>
<tr>
<td>Gear</td>
<td>Int</td>
<td>-</td>
<td>Current gear</td>
</tr>
</tbody>
</table>

4.3.2 SoftECU

*Inputs*

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starter</td>
<td>Bool</td>
<td>-</td>
<td>Received starter signal</td>
</tr>
<tr>
<td>n_Engine</td>
<td>Double</td>
<td>rpm</td>
<td>Measured engine speed</td>
</tr>
<tr>
<td>M_Ind_In</td>
<td>Double</td>
<td>Nm</td>
<td>Measured induced engine torque</td>
</tr>
<tr>
<td>M_Drag_In</td>
<td>Double</td>
<td>Nm</td>
<td>Measured engine drag torque</td>
</tr>
<tr>
<td>M_CombEngine</td>
<td>Double</td>
<td>Nm</td>
<td>Effective engine torque</td>
</tr>
<tr>
<td>N_ThrottlePlateOpen-ing_r_0_100</td>
<td>Double</td>
<td>%</td>
<td>Measured throttle plate position</td>
</tr>
<tr>
<td>In_Gearbox_Shifting</td>
<td>Bool</td>
<td>-</td>
<td>Flag indicating active gearbox shifting</td>
</tr>
<tr>
<td>Ignition</td>
<td>Bool</td>
<td>-</td>
<td>Driver ignition flag</td>
</tr>
<tr>
<td>Gear</td>
<td>Int</td>
<td>-</td>
<td>Current gear</td>
</tr>
</tbody>
</table>
4.3.3 Vehicle/Drivetrain

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CU_Gear</td>
<td>Int</td>
<td>-</td>
<td>Requested gear</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual_Gear</td>
<td>Int</td>
<td>-</td>
<td>Current gear</td>
</tr>
<tr>
<td>GearboxShifting</td>
<td>Bool</td>
<td>-</td>
<td>Flag indicating active gear box shifting</td>
</tr>
<tr>
<td>n_Engine</td>
<td>Double</td>
<td>rpm</td>
<td>Calculated engine speed</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerator_Out</td>
<td>Double</td>
<td>-</td>
<td>Processed accelerator pedal position</td>
</tr>
<tr>
<td>Brake_Out</td>
<td>Double</td>
<td>-</td>
<td>Processed brake pedal position</td>
</tr>
<tr>
<td>Clutch_Out</td>
<td>Double</td>
<td>-</td>
<td>Processed clutch pedal position</td>
</tr>
<tr>
<td>Gear_Out</td>
<td>Int</td>
<td>-</td>
<td>Processed gear</td>
</tr>
<tr>
<td>Ignition_Out</td>
<td>Bool</td>
<td>-</td>
<td>Processed ignition flag</td>
</tr>
<tr>
<td>M_Target_fast_Out</td>
<td>Double</td>
<td>Nm</td>
<td>Desired torque for fast requests</td>
</tr>
<tr>
<td>M_Target_slow_Out</td>
<td>Double</td>
<td>Nm</td>
<td>Desired torque for slow requests</td>
</tr>
<tr>
<td>Starter_Out</td>
<td>Bool</td>
<td>-</td>
<td>Processed starter flag</td>
</tr>
<tr>
<td>SignalBit_slow_out</td>
<td>Bool</td>
<td>-</td>
<td>Flag indicating fast torque request</td>
</tr>
<tr>
<td>SignalBit_fast_out</td>
<td>Bool</td>
<td>-</td>
<td>Flag indicating slow torque request</td>
</tr>
</tbody>
</table>
4.3.4 Vehicle/CombustionEngine

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SignalBit_slow</td>
<td>Bool</td>
<td>-</td>
<td>Flag indicating slow torque request</td>
</tr>
<tr>
<td>SignalBit_fast</td>
<td>Bool</td>
<td>-</td>
<td>Flag indicating fast torque request</td>
</tr>
<tr>
<td>M_Target_slow</td>
<td>Double</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_Target_fast</td>
<td>Double</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_Drag</td>
<td>Double</td>
<td>Nm</td>
<td>Calculated engine drag torque</td>
</tr>
<tr>
<td>M_Engine</td>
<td>Double</td>
<td>Nm</td>
<td>Calculated engine torque</td>
</tr>
<tr>
<td>M_Ind</td>
<td>Double</td>
<td>Nm</td>
<td>Calculated induced engine torque</td>
</tr>
<tr>
<td>M_Ind_Max</td>
<td>Double</td>
<td>Nm</td>
<td>Calculated maximum induced engine torque</td>
</tr>
<tr>
<td>ThrottlePlateOpen-</td>
<td>Double</td>
<td>-</td>
<td>Throttle plate opening</td>
</tr>
<tr>
<td>ing_r_0_100</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.3.5 Driver/BodyComponents

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CU_Starter</td>
<td>Bool</td>
<td>-</td>
<td>Starter status communicated by CU</td>
</tr>
<tr>
<td>CU_Ignition</td>
<td>Bool</td>
<td>-</td>
<td>Ignition status communicated by CU</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BatteryIsOn</td>
<td>Bool</td>
<td>-</td>
<td>Battery status</td>
</tr>
</tbody>
</table>

**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 four sections in the following figure:

1. 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 chapter "Driver Modes" on page 39. After selecting the driver mode, you can control the vehicle by interacting with the available controls. The individual control elements are described in the chapter "Controlling the Vehicle" on page 42.

2. "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.

3. 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.

4. The section "Simulation" contains graphs of the engine speed and the vehicle speed over time.
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 chapter "Project and Experiment Creation" on page 19.

### 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:

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 42.

#### Target Speed - Internal Gear (MT)

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)
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
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

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

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

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

Usage of these two controls is special and explained in the chapter “Driving Test Cycles” on page 44.

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/Derive 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:

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 chapter “Getting Started with the Tutorial Project” on page 28 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, a step-by-step guide is given, which describes the generation of 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 VVTB_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 LABCAR-OPERATOR

This chapter 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 chapter.

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:

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\_Lib\VVTB_MT_ICE\mdl\Simulink_VVTB\src.
5. Click Finish.

-> The Project Explorer shows the added module.
To add the internal combustion engine model
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\_Lib\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
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\_Lib\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
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\_Lib\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\_Lib\VVTB_MT_ICE\conn" and at ".\ICE\Project Items\_Lib\ICE_G_1B_1Stg_DEMO\conn."

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

**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 "VVTB_ICE_LC" module.

3. Click **Build**.

![Build window](image)

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.

### 5.1.2 Experiment Creation with LABCAR-OPERATOR

After the successful build of the project, 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-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\_Lib\VVTB_MT_ICE\mapping.

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\_Lib\ICE_G_1B_1Stg_DEMO\param\0001_Initialization.

NOTE

Ensure that the added mapping 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 \VVVB\Project_Items\_Lib\VVVB_MT\rtp\Driver_Instruments.

NOTE

Ensure that the added parameter files are also activated.
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.
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\_Lib\VVTB_MT_ICE\img.
2. Create the target folder if it does not already exist.
3. Import the Driver.layer file as located in .\VVTB\Project Items\_Lib\VVTB_MT_ICE\gui.

-> After the successful import of the Driver layer, basic dashboards for handling the vehicle model exist.
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 chapter "Getting Started with the Tutorial Project" on page 56 describes the handling of the simulation model in more detail.

**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. The layer can be adapted 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 chapter "Getting Started with the Tutorial Project" on page 56 describes the handling of the simulation model in more detail.
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 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.

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 38 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\Lib\ICE_G_1B_1Stg_DEMO\gui.

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 Soft-ECU 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.
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.

**Air Mass Flow Manipulation in Manual Mode**

To activate the manual mode, do the following in the field "Simulation Control Parameters":

1. Set the parameter
   \[ \text{Throttle\_Area\_Calc\_DEVM\_1\_GEVM\_0/Value} \rightarrow 1 \]
2. Set the parameter \[ \text{A\_Throttle\_Area\_Mode} \rightarrow \text{Manual Mode} \]
3. Set the parameter \[ \text{A\_Throttle\_Area\_Manual} \rightarrow 0.5 \]

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.
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.

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.
5.3 Signal Interface

The basic interface to operate LABCAR-MODEL-ICE, as used in the chapter “Getting Started with the Tutorial Project” on page 56, 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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>phi_CAModel</td>
<td>Double</td>
<td>CA</td>
<td>Crankshaft angle</td>
</tr>
<tr>
<td>FlywheelStandstill</td>
<td>Bool</td>
<td>-</td>
<td>Engine operation status</td>
</tr>
<tr>
<td>Ignition</td>
<td>Bool</td>
<td>-</td>
<td>Ignition status</td>
</tr>
<tr>
<td>p_Ambient</td>
<td>Double</td>
<td>Pa</td>
<td>Ambient pressure</td>
</tr>
<tr>
<td>T_Ambient</td>
<td>Double</td>
<td>K</td>
<td>Ambient temperature</td>
</tr>
<tr>
<td>Accelerator</td>
<td>Double</td>
<td>-</td>
<td>Accelerator position</td>
</tr>
<tr>
<td>n_Engine</td>
<td>Double</td>
<td>rpm</td>
<td>Engine speed</td>
</tr>
</tbody>
</table>

[Diagram of engine model with inputs and interfaces]
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U_Batt</td>
<td>Double</td>
<td>V</td>
<td>Battery voltage</td>
</tr>
<tr>
<td>M_Engage</td>
<td>Double</td>
<td>Nm</td>
<td>Torque of auxiliary systems</td>
</tr>
<tr>
<td>v_Vehicle</td>
<td>Double</td>
<td>km/h</td>
<td>Vehicle speed</td>
</tr>
<tr>
<td>v_Distance</td>
<td>Double</td>
<td>m</td>
<td>Traveled distance of vehicle</td>
</tr>
<tr>
<td>Clutch</td>
<td>Double</td>
<td></td>
<td>Clutch pedal position</td>
</tr>
<tr>
<td>Gear</td>
<td>Int</td>
<td></td>
<td>Gear position</td>
</tr>
<tr>
<td>M_Engine</td>
<td>Double</td>
<td>Nm</td>
<td>Engine torque</td>
</tr>
<tr>
<td>J_Engine</td>
<td>Double</td>
<td>kgm²</td>
<td>Moment of inertia of engine</td>
</tr>
<tr>
<td>M_EngineStick</td>
<td>Double</td>
<td>Nm</td>
<td>Engine stick torque of resting engine</td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_Engine</td>
<td>Double</td>
<td>Nm</td>
<td>Engine torque</td>
</tr>
<tr>
<td>J_Engine</td>
<td>Double</td>
<td>kgm²</td>
<td>Moment of inertia of engine</td>
</tr>
<tr>
<td>M_EngineStick</td>
<td>Double</td>
<td>Nm</td>
<td>Engine stick torque of resting engine</td>
</tr>
</tbody>
</table>

### 5.3.2 Combustion Engine/Injection

#### Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_Injection1</td>
<td>Double</td>
<td>µs</td>
<td>Injection time cylinder 1</td>
</tr>
<tr>
<td>t_Injection2</td>
<td>Double</td>
<td>µs</td>
<td>Injection time cylinder 2</td>
</tr>
<tr>
<td>t_Injection3</td>
<td>Double</td>
<td>µs</td>
<td>Injection time cylinder 3</td>
</tr>
<tr>
<td>t_Injection4</td>
<td>Double</td>
<td>µs</td>
<td>Injection time cylinder 4</td>
</tr>
<tr>
<td>t_Injection5</td>
<td>Double</td>
<td>µs</td>
<td>Injection time cylinder 5</td>
</tr>
<tr>
<td>t_Injection6</td>
<td>Double</td>
<td>µs</td>
<td>Injection time cylinder 6</td>
</tr>
</tbody>
</table>

### 5.3.3 Combustion Engine/Lambda Sensor

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lambda</td>
<td>Double</td>
<td></td>
<td>Lambda sensor before catalyst</td>
</tr>
</tbody>
</table>
5.3.4 Combustion Engine/Throttle Plate

*Inputs*

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC_Cannel_0_1</td>
<td>Double</td>
<td>-</td>
<td>Opening value throttle plate</td>
</tr>
</tbody>
</table>

5.3.5 Combustion Engine/Engine Torque

*Outputs*

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_EngineOut_Nm</td>
<td>Double</td>
<td>Nm</td>
<td>Effective drive torque</td>
</tr>
<tr>
<td>M_Ind_Nm</td>
<td>Double</td>
<td>Nm</td>
<td>Indicated engine torque</td>
</tr>
<tr>
<td>M_Drag_Nm</td>
<td>Double</td>
<td>Nm</td>
<td>Engine drag torque</td>
</tr>
</tbody>
</table>
6 LABCAR-MODEL-PMSM

ETAS LABCAR-MODEL-PMSM is a 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. It is an FPGA-based model so that low latency small cycle times are possible.

6.1 Overview

The LABCAR-MODEL-PMSM is implemented to use the FPGA of the ES5340. Therefore the LABCAR-MODEL-PMSM is an FPGA simulation model, which has been implemented using the ETAS tool FlexibleFPGA V3.0.0 and can run on this specific ETAS hardware board. For detailed information see [3].

The LABCAR-MODEL-PMSM supports the 8 analog outputs of the ES5340.2-M board. The slave board is not used. (The ES5340.2-M acts as a master board. It can be connected to the ES5340.1-S slave board to offer more analog and digital output channels. This connection can be configured inside an ETAS RTPC. For more detailed information, see the ES5340.2 Electric Drive Simulation Board User’s Guide.)

LABCAR-MODEL-PMSM exists in two variants, Basic and Advanced. The Basic variant consists of three main components: an ideal inverter, a PMSM e-motor, and a simplistic drivetrain model. In addition, the Advanced variant offers some more parameters. These are iron losses in the e-motor and ohmic resistances in the inverter.

6.2 Project and Experiment Creation

This chapter describes flashing LABCAR-MODEL-PMSM to ES5340.2-M board and creating a LABCAR-OPERATOR project.

6.2.1 Flashing LABCAR-MODEL-PMSM to ES5340.2-M

This section describes the procedure for flashing the LABCAR-MODEL-PMSM to the ES5340.2-M. The prerequisite for using a LABCAR-OPERATOR project is flashing the model to the ES5340.2-M. It is required to ensure that the ES5340.2-M board is inserted to the RTPC and the host PC is connected with the RTPC.

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.

The following two steps are required to program the LABCAR-MODEL-PMSM to the FPGA.

- Run the FlexFPGACopy.exe file.
  You find more details on this tool in [4].
• Use the "HSP Update Tool" to program or flash the model.

The HSP Update Tool enables an easy firmware update of ETAS hardware products. You find the HSP Update Tool on the ETAS Software CD. More details can be found in [6].

To program the LABCAR-MODEL-PMSM to the ES5340, do the following:

1. Run "FlexFPGACopy.exe".
   The tool copies the PMSM model file ("ES5340_2MFFGA67109121.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

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**.

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

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

   -> After the successful update, the below window appears.

9. Click **OK**.

   The installed version (model version) of the "Component" "FlexibleFPGA Support" 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 "PMSM_Basic" in a simple test environment.
This chapter 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-PMSM with LABCAR-OPERATOR, follow the steps described in this chapter.

As a preliminary step, you successfully flashed the latest LABCAR-MODEL-PMSM software to the ES5340.2-M board as described in chapter "Flashing LABCAR-MODEL-PMSM to ES5340.2-M" on page 64.

To set up the LABCAR-OPERATOR project, do the following:

1. Open LABCAR-OPERATOR
2. Create a new Project called "PMSM_Basic".
   - After creation, the LABCAR-OPERATOR project environment looks as follows:

   ![Image of LABCAR-OPERATOR project environment]

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

   To integrate the LABCAR-MODEL-PMSM model, do the following:

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

   ![Image of Project Explorer]

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

![Add Item Window]

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

6. Click **Ok**.

![Add Item Window]

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".

   -> Now the LABCAR-MODEL-PMSM appears on LABCAR-RTC.
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 LABCAR-MODEL-PMSM to ES5340.2-M" on page 64.

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

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 generation of current maps is described in the chapter "Nonlinear Maps and Parameter Maps Downloader" on page 81.

**To integrate the Simulink module, do the following:**

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\Lib\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 temperate effect of coils.

To add a C-code module, do the following:
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\Lib\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 FlexibleFPGAModel. Hence, connection files are used to connect the SIMULINK module and the C-Code module with the FlexibleFPGAModel.

To connect the modules, do the following:

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

![Connection Manager](image)

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\_Lib\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\Lib\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:

   ![Image of Connection Manager](image)

### 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, do the following:**

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”.

6. In the “Task” window, select “TaskDVMMModel”.

7. Set the period to 0.0001.

8. In the “Task” window, select “TaskDVEModel_Parameter_Download”.

9. Set the period to 0.0004.

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, do the following:**

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

The build process performs the following basic steps:

- Simulink code generation on host pc
- Transfer of sources to RTPC
- Compilation of sources on RTPC
- Transfer of binaries back to host pc

### 6.2.3 Experiment Creation with LABCAR-OPERATOR

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 LABCAR-MODEL-PMSM to ES5340.2-M” on page 64 is the same as the model version that is used in the section “Project Creation in LABCAR-OPERATOR” on page 66.
To start LABCAR-EE and to access the FlexibleFPGA ports:

1. Open LABCAR-EE using LABCAR-IP

Tools → Open Experiment Environment

- The Experiment Environment opens.

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 the FlexibleFPGAModel 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_01_Simulation_Control.layer
- PMSM_Failure_Simulations.layer
- PMSM_Parameters_Ideal.layer
- PMSM_Runtime_License. Layer
You find the layers in the install folder under \Project Items\_Lib\PMSM_Param_Download\gui.

To add layers, do the following:
1. Right click on Instrumentation Layer.
2. Click Import Layer to add layers.

![Layer Import](image)

**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:

![Experimental Environment](image)

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 LABCAR-MODEL-PMSM to ES5340.2-M” on page 64. The chapter “Getting Started with Tutorial Project” on page 78 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 LABCAR-MODEL-PMSM to ES5340.2-M" on page 64.

The tutorial project is located in \Example Project 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.

### 6.3.1 Overview

The prerequisite for using the LABCAR-OPERATOR project and the PMSM license is that the model version installed to the ES5340.2-M, as described in the section 6.2.1, is identical to the model version (*.adf) as described in section 6.2.2. Use LABCAR-EE to parametrize the model.

### 6.3.2 Active Short-Circuit Operation Mode

LABCAR-MODEL-PMSM is operated in active short condition. This test operating mode does not require any PWM signal actuation to the model. The machine can be provided 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.
2. Use the "PMSM Simulation Control" GUI Layer for setting this input.

   ![Inverter GUI](image)

   ![DC Bus Voltage](image)

   The intermediate circuit flag can be set to "Dis-connected" in "PMSM Parameters" layer. Thus the DC bus voltage is the same as the battery voltage.
3. 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.
4. Set sample test parameters as shown below

   ![PMSM Parameters](image)

   ![Stator Coil Resistance](image)
5. 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.

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

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., $L_d$ and $L_q$ as $f(I_d, I_q)$. 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 $I_d(\Phi_d, \Phi_q)$ and $I_q(\Phi_d, \Phi_q)$ are required.

Here

$L_d$ and $L_q$ are inductance [H] along d and q axis respectively.
$I_q$ and $I_q$ are currents [A] along d and q axis respectively.
$\Phi_d$ and $\Phi_q$ are fluxes [Vs] along d and q axis respectively.

The chapter 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 ESS340.2-M.

A MATLAB based tool "CurrentTablesGenerationTool.m" is provided to invert the inductance tables ($f(I_d, I_q)$) to obtain lookup tables for $I_d$ and $I_q$ ($f(\Phi_d$ and $\Phi_q)$).

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 ($\Phi_d$) and q-direction ($\Phi_q$) 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 ($\Phi_d$) 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 51200 (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 chapter "Project and Experiment Creation" on page 64 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 $\varphi_d$ and $\varphi_q$.

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

LABCAR-MODEL-PMSM contains a model for an inverter, a model for a PMSM motor, and a model for a drivetrain. In the following, the basic interfaces to operate LABCAR-MODEL-PMSM are given.

#### 6.5.1 Drivetrain Interfaces

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_L</td>
<td>fixdt(1,32,21)</td>
<td>Nm</td>
<td>load torque</td>
</tr>
<tr>
<td>Phi_mech_Init</td>
<td>fixdt(0,32,28)</td>
<td>rad</td>
<td>Initial mechanical rotor angle</td>
</tr>
<tr>
<td>Omega_Input</td>
<td>fixdt(1,32,15)</td>
<td>rad/s</td>
<td>Speed input of drivetrain in speed mode</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Omega_mech</td>
<td>fixdt(1,32,15)</td>
<td>rad/s</td>
<td>mechanical rotor velocity</td>
</tr>
<tr>
<td>Omega_el</td>
<td>fixdt(1,32,15)</td>
<td>rad/s</td>
<td>Electrical rotor velocity</td>
</tr>
<tr>
<td>Phi_mech_rad</td>
<td>fixdt(0,18,15)</td>
<td>rad</td>
<td>Mechanical rotor angle</td>
</tr>
<tr>
<td>Phi_el</td>
<td>fixdt(1,18,15)</td>
<td>rad</td>
<td>Electrical rotor angle</td>
</tr>
<tr>
<td>T_mech</td>
<td>fixdt(1,32,21)</td>
<td>Nm</td>
<td>Resulting mechanical rotor torque</td>
</tr>
</tbody>
</table>
6.5.2 Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TeL_average</td>
<td>fixdt(1,32,18)</td>
<td>Nm</td>
<td>Average electrical torque</td>
</tr>
<tr>
<td>Phi_mech_Degree</td>
<td>Int(16)</td>
<td>deg</td>
<td>Mechanical rotor angle</td>
</tr>
<tr>
<td>RPM</td>
<td>Int(32)</td>
<td></td>
<td>Revolutions per minute</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>fixdt(0,32,28)</td>
<td>Nms/ rad</td>
<td>Friction coefficient of E-machine; to deactivate set parameter to zero value</td>
</tr>
<tr>
<td>C</td>
<td>fixdt(0,32,8)</td>
<td>Nm/ rad</td>
<td>Torsional stiffness between rotor and load; to deactivate set parameter to zero value</td>
</tr>
<tr>
<td>D</td>
<td>fixdt(0,32,22)</td>
<td>Nms/ rad</td>
<td>Damping between rotor and load; to deactivate set parameter to zero value</td>
</tr>
<tr>
<td>Direct Feed through</td>
<td>Boolean</td>
<td>-</td>
<td>mode selector 0: use torque input 1: use speed input</td>
</tr>
<tr>
<td>invJ_E</td>
<td>fixdt(0,32,12)</td>
<td>1/ kgm2</td>
<td>Inverse of inertia of rotor</td>
</tr>
<tr>
<td>invJ_L</td>
<td>fixdt(0,32,12)</td>
<td>1/ kgm2</td>
<td>Inverse of inertia of load/ drivetrain</td>
</tr>
</tbody>
</table>

6.5.3 Inverter Interfaces

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWM_UH</td>
<td>Boolean</td>
<td>-</td>
<td>Control signal upper IGBT phase U</td>
</tr>
<tr>
<td>PWM_UL</td>
<td>Boolean</td>
<td>-</td>
<td>Control signal lower IGBT phase U</td>
</tr>
<tr>
<td>PWM_VH</td>
<td>Boolean</td>
<td>-</td>
<td>Control signal upper IGBT phase V</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------</td>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PWM_VL</td>
<td>Boolean</td>
<td>-</td>
<td>Control signal lower IGBT phase V</td>
</tr>
<tr>
<td>PWM_WH</td>
<td>Boolean</td>
<td>-</td>
<td>Control signal upper IGBT phase W</td>
</tr>
<tr>
<td>PWM_WL</td>
<td>Boolean</td>
<td>-</td>
<td>Control signal lower IGBT phase W</td>
</tr>
<tr>
<td>Gate_Signal_Polarity</td>
<td>Boolean</td>
<td>-</td>
<td>Polarity control 0: Low active 1: High active &amp;1</td>
</tr>
<tr>
<td>Fail_IGBT_HS_U</td>
<td>Fixdt(0,2,0)</td>
<td>-</td>
<td>Gate failure mode of high side switch of phase U 0: no error insertion 1: stuck to low 2: stuck to high</td>
</tr>
<tr>
<td>Fail_IGBT_HL_U</td>
<td>Fixdt(0,2,0)</td>
<td>-</td>
<td>Gate failure mode of low side switch of phase U 0: no error insertion 1: stuck to low 2: stuck to high</td>
</tr>
<tr>
<td>Fail_IGBT_HS_V</td>
<td>Fixdt(0,2,0)</td>
<td>-</td>
<td>Gate failure mode of low side switch of phase V 0: no error insertion 1: stuck to low 2: stuck to high</td>
</tr>
<tr>
<td>Fail_IGBT_HL_V</td>
<td>Fixdt(0,2,0)</td>
<td>-</td>
<td>Gate failure mode of high side switch of phase V 0: no error insertion 1: stuck to low 2: stuck to high</td>
</tr>
<tr>
<td>Fail_IGBT_HS_W</td>
<td>Fixdt(0,2,0)</td>
<td>-</td>
<td>Gate failure mode of high side switch of phase W 0: no error insertion 1: stuck to low 2: stuck to high</td>
</tr>
<tr>
<td>Fail_IGBT_HL_W</td>
<td>Fixdt(0,2,0)</td>
<td>-</td>
<td>Gate failure mode of low side switch of phase W 0: no error insertion 1: stuck to low 2: stuck to high</td>
</tr>
<tr>
<td>Fail_Diode_HS_U</td>
<td>Fixdt(0,2,0)</td>
<td>-</td>
<td>Diode failure mode of high side switch of phase U 0: no error insertion 1: stuck to low 2: stuck to high</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------</td>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Fail_Diode_HL_U</td>
<td>Fixdt(0,2,0)</td>
<td>-</td>
<td>Diode failure mode of low side switch of phase U</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: no error insertion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: stuck to low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: stuck to high</td>
</tr>
<tr>
<td>Fail_Diode_HS_V</td>
<td>Fixdt(0,2,0)</td>
<td>-</td>
<td>Diode failure mode of high side switch of phase V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: no error insertion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: stuck to low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: stuck to high</td>
</tr>
<tr>
<td>Fail_Diode_HL_V</td>
<td>Fixdt(0,2,0)</td>
<td>-</td>
<td>Diode failure mode of low side switch of phase V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: no error insertion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: stuck to low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: stuck to high</td>
</tr>
<tr>
<td>Fail_Diode_HS_W</td>
<td>Fixdt(0,2,0)</td>
<td>-</td>
<td>Diode failure mode of high side switch of phase W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: no error insertion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: stuck to low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: stuck to high</td>
</tr>
<tr>
<td>Fail_Diode_HL_W</td>
<td>Fixdt(0,2,0)</td>
<td>-</td>
<td>Diode failure mode of low side switch of phase W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: no error insertion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: stuck to low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: stuck to high</td>
</tr>
<tr>
<td>U_Batt0</td>
<td>Fixdt(0,16,5)</td>
<td>V</td>
<td>Input voltage of battery</td>
</tr>
<tr>
<td>RC_Ckt_Disable_Flag</td>
<td>Boolean</td>
<td>-</td>
<td>RC circuit/Intermediate circuit disable flag of Inverter 0: Normal 1: RC Circuit would be disabled</td>
</tr>
<tr>
<td>Relay_Switch</td>
<td>Boolean</td>
<td>-</td>
<td>Switch to connect/disconnect to battery 0: Not connects to battery 1: Connect to battery</td>
</tr>
<tr>
<td>ShortCircuit_Flag_Reset</td>
<td>Boolean</td>
<td>-</td>
<td>Input for resetting short circuit error flags</td>
</tr>
</tbody>
</table>
### Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U_DC</td>
<td>fixdt(1,32,18)</td>
<td>V</td>
<td>Intermediate circuit voltage</td>
</tr>
<tr>
<td>I_Batt</td>
<td>fixdt(1,32,18)</td>
<td>A</td>
<td>Battery current</td>
</tr>
<tr>
<td>I_DC</td>
<td>fixdt(1,18,4)</td>
<td>A</td>
<td>Intermediate circuit current</td>
</tr>
<tr>
<td>U_phase_U</td>
<td>fixdt(1,32,18)</td>
<td>V</td>
<td>Voltage Phase U</td>
</tr>
<tr>
<td>U_phase_V</td>
<td>fixdt(1,32,18)</td>
<td>V</td>
<td>Voltage Phase V</td>
</tr>
<tr>
<td>U_phase_W</td>
<td>fixdt(1,32,18)</td>
<td>V</td>
<td>Voltage Phase W</td>
</tr>
<tr>
<td>ShortCircuit_Error_-Flag_U</td>
<td>Boolean</td>
<td>-</td>
<td>Short circuit error flag for half bridge phase U</td>
</tr>
<tr>
<td>ShortCircuit_Error_-Flag_V</td>
<td>Boolean</td>
<td>-</td>
<td>Short circuit error flag for half bridge phase V</td>
</tr>
<tr>
<td>ShortCircuit_Error_-Flag_W</td>
<td>Boolean</td>
<td>-</td>
<td>Short circuit error flag for half bridge Phase W</td>
</tr>
<tr>
<td>I_element_H_U</td>
<td>fixdt(1,16,4)</td>
<td>A</td>
<td>Current through upper switch phase U</td>
</tr>
<tr>
<td>I_element_H_V</td>
<td>fixdt(1,16,4)</td>
<td>A</td>
<td>Current through upper switch phase V</td>
</tr>
<tr>
<td>I_element_H_W</td>
<td>fixdt(1,16,4)</td>
<td>A</td>
<td>Current through upper switch phase W</td>
</tr>
<tr>
<td>I_element_L_U</td>
<td>fixdt(1,16,4)</td>
<td>A</td>
<td>Current through lower switch phase U</td>
</tr>
<tr>
<td>I_element_L_V</td>
<td>fixdt(1,16,4)</td>
<td>A</td>
<td>Current through lower switch phase V</td>
</tr>
<tr>
<td>I_element_L_W</td>
<td>fixdt(1,16,4)</td>
<td>A</td>
<td>Current through lower switch phase W</td>
</tr>
<tr>
<td>U_Switch_H_U</td>
<td>fixdt(1,32,18)</td>
<td>V</td>
<td>Voltage on upper switch phase U</td>
</tr>
<tr>
<td>U_Switch_H_V</td>
<td>fixdt(1,32,18)</td>
<td>V</td>
<td>Voltage on upper switch phase V</td>
</tr>
<tr>
<td>U_Switch_H_W</td>
<td>fixdt(1,32,18)</td>
<td>V</td>
<td>Voltage on upper switch phase W</td>
</tr>
<tr>
<td>U_Switch_L_U</td>
<td>fixdt(1,32,18)</td>
<td>V</td>
<td>Voltage on lower switch phase U</td>
</tr>
<tr>
<td>U_Switch_L_V</td>
<td>fixdt(1,32,18)</td>
<td>V</td>
<td>Voltage on lower switch phase V</td>
</tr>
<tr>
<td>U_Switch_L_W</td>
<td>fixdt(1,32,18)</td>
<td>V</td>
<td>Voltage on lower switch phase W</td>
</tr>
</tbody>
</table>
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocked_U</td>
<td>Boolean</td>
<td>-</td>
<td>Flag sets if both upper and lower switches of</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOCKED_V</td>
<td>Boolean</td>
<td>-</td>
<td>Flag sets if both upper and lower switches of</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOCKED_W</td>
<td>Boolean</td>
<td>-</td>
<td>Flag sets if both upper and lower switches of</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Name**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>U_D_FWD</td>
<td>fixdt(0,16,14)</td>
<td>V</td>
<td>Conducting voltage in Diode (293.15K)</td>
<td>0.2</td>
<td>0.01...3</td>
</tr>
<tr>
<td>Inv_R_Batt</td>
<td>fixdt(0,16,12)</td>
<td>1/Ω</td>
<td>Inverse of inner resistance of battery</td>
<td>0.5</td>
<td>1e-3...10</td>
</tr>
<tr>
<td>Inv_C_DC</td>
<td>fixdt(0,16,12)</td>
<td>1/µF</td>
<td>Inverse of intermediate circuit capacitor</td>
<td>0.1</td>
<td>1e-3...0.1</td>
</tr>
<tr>
<td>T_Switch 1</td>
<td>fixdt(0,16,32)</td>
<td>s</td>
<td>Switching time of IGBT</td>
<td>5e-6</td>
<td>0.2e-6...10e-6</td>
</tr>
<tr>
<td>R_SW_ON 1</td>
<td>fixdt(0,16,20)</td>
<td>Ω</td>
<td>Conducting resistance in IGBT</td>
<td>0.001</td>
<td>0.0...0.05</td>
</tr>
<tr>
<td>R_D_FWD 1</td>
<td>fixdt(0,16,20)</td>
<td>Ω</td>
<td>Conducting resistance in Diode</td>
<td>0.001</td>
<td>0.0...0.05</td>
</tr>
<tr>
<td>U_SW_FWD 1</td>
<td>fixdt(0,16,14)</td>
<td>V</td>
<td>Conducting voltage in IGBT (293.15K)</td>
<td>0.2</td>
<td>0.01...3</td>
</tr>
</tbody>
</table>

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

**NOTE**

1e6 is multiplied inside FPGA so the input for Inv_C_DC shall be 1/C_Actual. Where C_Actual is the actual capacitance of the Intermediate circuit capacitor (µF).

**NOTE**

The time constant of the intermediate circuit (R_Batt * C_DC) must be greater than 1µs.
### 6.5.4 PMSM Motor Interfaces

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rs</td>
<td>Single</td>
<td>Ω</td>
<td>Stator coil resistance</td>
</tr>
<tr>
<td>Riron (^2)</td>
<td>Single</td>
<td>Ω</td>
<td>Resistance represents iron losses of machine</td>
</tr>
<tr>
<td>L(_d)</td>
<td>Single</td>
<td>H</td>
<td>Inductance in d axis</td>
</tr>
<tr>
<td>L(_q)</td>
<td>Single</td>
<td>H</td>
<td>Inductance in q axis</td>
</tr>
<tr>
<td>Psi_Rotor</td>
<td>Single</td>
<td>Vs</td>
<td>Permanent Magnetic flux in rotor</td>
</tr>
<tr>
<td>Linear_NonLinear_Selector</td>
<td>Boolean</td>
<td></td>
<td>0: linear mode (Inputs L(_d) and L(_q) would be considered)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Non-linear mode (loaded parameter fields to FPGA Id and IQ f(Psi(_d),Psi(_q)) to FPGA would be considered)</td>
</tr>
<tr>
<td>Current_DQ_Selector</td>
<td>Boolean</td>
<td></td>
<td>0: Internal calculated values for Id and IQ would be considered</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Stimulus inputs, ID_Stimulate and IQ_Stimulate would be considered</td>
</tr>
<tr>
<td>ID_Stimulate</td>
<td>Single</td>
<td>A</td>
<td>Stimulating input for Id</td>
</tr>
<tr>
<td>IQ_Stimulate</td>
<td>Single</td>
<td>A</td>
<td>Stimulating input for IQ</td>
</tr>
<tr>
<td>Reset</td>
<td>Boolean</td>
<td></td>
<td>Input for resetting internal states of model</td>
</tr>
<tr>
<td>AngleCorrectionSamples</td>
<td>fixdt(0,8,2)</td>
<td></td>
<td>Number of samples considered for angle correction of Id and IQ calculations</td>
</tr>
<tr>
<td>FixPointOverflow_Reset</td>
<td>Boolean</td>
<td></td>
<td>Input to clear fix point overflow state</td>
</tr>
</tbody>
</table>

\(^2\) This signal is only available in LABCAR-MODEL-PMSM Advanced.

---

**NOTE**

“FixPointOverflow_Status” and “FixPointOverflow_Reset” are used for advanced support.
6.5.5 DAC Interfaces

The LABCAR-MODEL-PMSM supports the 8 analog outputs of the ES5340.2-M board. You can configure the outputs. 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 one of the signals of the simulation model by using

### Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vd</td>
<td>Single</td>
<td>V</td>
<td>Voltage in d axis</td>
</tr>
<tr>
<td>Vq</td>
<td>Single</td>
<td>V</td>
<td>Voltage in q axis</td>
</tr>
<tr>
<td>Psi_d</td>
<td>Single</td>
<td>V</td>
<td>Flux in d axis</td>
</tr>
<tr>
<td>Psi_q</td>
<td>Single</td>
<td>V</td>
<td>Flux in q axis</td>
</tr>
<tr>
<td>Torque</td>
<td>Single</td>
<td>Nm</td>
<td>Motor Electromagnetic Torque</td>
</tr>
<tr>
<td>I_Stator_X(X=U,V,W)</td>
<td>Single</td>
<td>A</td>
<td>Stator currents</td>
</tr>
<tr>
<td>U_Ind_X (X=U,V,W)</td>
<td>Single</td>
<td>V</td>
<td>Induced back EMF</td>
</tr>
<tr>
<td>Id</td>
<td>Single</td>
<td>A</td>
<td>Current in d axis</td>
</tr>
<tr>
<td>Iq</td>
<td>Single</td>
<td>A</td>
<td>Current in q axis</td>
</tr>
<tr>
<td>PLoss_CopperId</td>
<td>Single</td>
<td>W</td>
<td>Copper loss of Motor</td>
</tr>
<tr>
<td>PLoss_Core</td>
<td>Single</td>
<td>W</td>
<td>Core loss of Motor</td>
</tr>
<tr>
<td>Icd</td>
<td>Single</td>
<td>A</td>
<td>Current in core loss branch of d axis</td>
</tr>
<tr>
<td>Icq</td>
<td>Single</td>
<td>A</td>
<td>Current in core loss branch q axis</td>
</tr>
<tr>
<td>Imd</td>
<td>Single</td>
<td>A</td>
<td>Current in magnetization branch of d axis</td>
</tr>
<tr>
<td>Imq</td>
<td>Single</td>
<td>A</td>
<td>Current in magnetization branch q axis</td>
</tr>
<tr>
<td>FixPointOverflow_Status</td>
<td>fixdt(0,45,0)</td>
<td>-</td>
<td>Debug Signal for fixed point overflow detection. This is used for internal purpose.</td>
</tr>
</tbody>
</table>

### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Description</th>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPN</td>
<td>Uint8</td>
<td>-</td>
<td>Number of pole pairs</td>
<td>1</td>
<td>1...32</td>
</tr>
</tbody>
</table>

**NOTE**

For a closed loop operation, LABCAR-MODEL-PMSM DAC and PWM digital input channel interfaces have to be connected.
the multiplexer as described in the table below. By choosing the multiplexer input number, a particular model signal can be connected to the analog output channel.

<table>
<thead>
<tr>
<th>Model Output Signal</th>
<th>DAC Multiplexer Input Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Omega_el</td>
<td>0</td>
</tr>
<tr>
<td>Omega_mech</td>
<td>1</td>
</tr>
<tr>
<td>Phi_el</td>
<td>2</td>
</tr>
<tr>
<td>Phi_mech_rad</td>
<td>3</td>
</tr>
<tr>
<td>T_mech</td>
<td>4</td>
</tr>
<tr>
<td>Tel_average</td>
<td>5</td>
</tr>
<tr>
<td>I_Phase_U</td>
<td>6</td>
</tr>
<tr>
<td>I_Phase_V</td>
<td>7</td>
</tr>
<tr>
<td>I_Phase_W</td>
<td>8</td>
</tr>
<tr>
<td>U_Phase_U</td>
<td>9</td>
</tr>
<tr>
<td>U_Phase_V</td>
<td>10</td>
</tr>
<tr>
<td>U_Phase_W</td>
<td>11</td>
</tr>
<tr>
<td>Id</td>
<td>12</td>
</tr>
<tr>
<td>Iq</td>
<td>13</td>
</tr>
<tr>
<td>U_Ind_U</td>
<td>14</td>
</tr>
<tr>
<td>U_Ind_V</td>
<td>15</td>
</tr>
<tr>
<td>U_Ind_W</td>
<td>16</td>
</tr>
<tr>
<td>Psi_d</td>
<td>17</td>
</tr>
<tr>
<td>Psi_q</td>
<td>18</td>
</tr>
<tr>
<td>U_DC</td>
<td>19</td>
</tr>
<tr>
<td>I_DC</td>
<td>20</td>
</tr>
<tr>
<td>Idc</td>
<td>21</td>
</tr>
<tr>
<td>Iqc</td>
<td>22</td>
</tr>
<tr>
<td>Imd</td>
<td>23</td>
</tr>
<tr>
<td>Imq</td>
<td>24</td>
</tr>
</tbody>
</table>

**NOTE**

To use analog outputs of the ES5340.2-M board, you have to add the configuration to the LABCAR-OPERATOR project using "hardware configuration". For more details on adding analog outputs to LABCAR-OPERATOR projects, refer to [4].
6.5.6 Digital Interfaces

LABCAR-MODEL-PMSM considers six "input gate control PWM of IGBT of inverter". The PWM digital input to the ESS340.2-M shall be added to the project using "hardware configuration". For more details on adding digital inputs to a LABCAR-OPERATOR project, please refer to [4].

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

![Digital Interface Diagram]

6.6 User Interfaces

The below mentioned Graphical User Interfaces (GUI) are provided for LABCAR-MODEL-PMSM, which contains basic functionality to operate the PMSM model comfortably. The GUI allows you to select various modes of the model, to enter PMSM license keys, and to obtain status.

LABCAR-MODEL-PMSM provides the following layers:

- 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.

If you enter correct license keys/codes the "license status" will be green, otherwise it will remain gray.
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" 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 chapter "Signal Interface" on page 82, 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 should be 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 voltage is (limited by HW to ±10V).
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:

$$\text{ScalingFactor} = \frac{\text{DesiredVoltageLevel}}{\text{SignalValue}}$$

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.

6.6.4 PMSM Failure Simulations

This GUI layer provides an interface to insert various fault (stuck to high or stuck to 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.

NOTE

To set the Offset to the signal voltage, use the model parameter "Offset_Anaout_X(X=0...6)".
• 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/diode is connected "1" always or conducting mode.
- Stuck to Low:
  The switch/diode is connected "0" always or open/non conducting.

• Short circuit indicator
For each phase of the inverter, a flag indicates if there 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 82.
The GUI uses standard instruments. This can easily be used as a template for individual GUI's.
7 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.

7.1 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".

7.1.1 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.

7.2 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].
7.2.1 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.

7.2.2 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 chapter 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

**EMS**
Engine Management system, i.e., the ECU that controls the combustion engine.

**FPGA**
Field Programmable Gate Array

**GUI**
Graphical User Interface

**HiL**
Hardware-in-the-Loop

**HSP**
Hardware Service Pack

**ICE**
Internal combustion engine

**IGBT**
Insulated Gate Bipolar transistor

**MCU**
Motor control unit, i.e., an ECU that controls an electrical motor.

**MiL**
Model-in-the-loop

**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 V5.4.9 User’s Guide
[2] LABCAR-OPERATOR V5.4.9 Software Compatibility List
# Figures

| Fig. 2-1 | General parametrization workflow | 11 |
| Fig. 2-2 | Online parametrization workflow | 12 |
| Fig. 2-3 | LCInports and LCOutports | 13 |
| Fig. 2-4 | Code generation process for Simulink models | 14 |
| Fig. 2-5 | Exemplary version information for an S-Function | 15 |
ETAS Contact Addresses

ETAS Headquarters
ETAS GmbH
Borsigstraße 24
70469 Stuttgart
Germany

Phone: +49 711 3423-0
Fax: +49 711 3423-2106
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