



Taking you to the next level



IPG Documentation

**CarMaker®**

User's Guide Version 5.1.4

The information in this document is furnished for informational use only, may be revised from time to time, and should not be construed as a commitment by IPG Automotive GmbH. IPG Automotive GmbH assumes no responsibility or liability for any errors or inaccuracies that may appear in this document.

This document contains proprietary and copyrighted information and may not be copied, reproduced, translated, or reduced to any electronic medium without prior consent, in writing, from IPG Automotive GmbH.

© 1999 - 2017 by IPG Automotive GmbH – [www.ipg-automotive.com](http://www.ipg-automotive.com)  
All rights reserved.

FailSafeTester, IPGCar, IPGControl, IPGDriver, IPGEngine, IPGGraph, IPGKinematics, IPGLock, IPGMotorcycle, IPGMovie, IPGRoad, IPGRoaddata, IPGTire, IPGTrailer, IPGTruck, RealtimeMaker, Xpack4 are trademarks of IPG Automotive GmbH.

CarMaker, TruckMaker, MotorcycleMaker, MESA VERDE are registered trademarks of IPG Automotive GmbH.

All other product names are trademarks of their respective companies.

# Contents

<b>1</b>	<b>About this User's Guide</b>	<b>13</b>
<b>2</b>	<b>General Overview</b>	<b>14</b>
2.1	System Description .....	14
2.1.1	The Virtual Vehicle Environment - VVE .....	14
2.1.2	The CarMaker Interface Toolbox - CIT .....	15
2.1.3	Real-Time vs. Office .....	16
2.1.4	CarMaker/Office .....	16
2.1.5	CarMaker/HIL .....	20
2.1.6	From MIL and SIL to HIL .....	23
2.2	Fields of Application .....	24
2.2.1	ECU Testing .....	24
2.2.2	Subsystem Testing .....	24
2.3	Summary .....	25
<b>3</b>	<b>Start CarMaker</b>	<b>26</b>
3.1	CarMaker/Office .....	26
3.2	CarMaker for Simulink .....	28
3.3	CarMaker/HIL .....	29
3.3.1	Remote Start for CarMaker/HIL Xeno .....	29
3.3.2	Manual Connecting to CarMaker/HIL Xeno .....	30
3.3.3	CarMaker on dSPACE Systems .....	31
3.3.4	CarMaker with ETAS Hardware .....	32
3.4	Directory Structure .....	33
3.4.1	Project Directory .....	33
3.4.2	Installation Directory .....	35

3.4.3	Shared Directory .....	35
<b>4</b>	<b>Parameterization: TestRun</b>	<b>37</b>
4.1	Definitions .....	37
4.2	Data Management .....	38
4.2.1	Load and Save .....	38
4.2.2	File System Overview .....	39
4.3	Starting from Scratch .....	39
4.4	IPGRoad .....	41
4.4.1	Overview of IPGRoad .....	41
4.4.2	General Settings .....	42
4.4.3	Segment Based Roads .....	44
4.4.4	Digitized Roads .....	47
4.4.5	IPGRoad 5.0: Road Networks .....	52
4.4.6	Checking Road Geometry .....	54
4.4.7	Adding Road Markers .....	58
4.4.8	Adding Road Bumps .....	65
4.4.9	Adding Sensor Objects .....	77
4.4.10	Adding Movie Geometries .....	82
4.4.11	Special Road Configurations .....	88
4.5	Maneuver .....	91
4.5.1	Overview .....	91
4.5.2	Building a Scenario - Creating a Maneuver list .....	92
4.5.3	Specification of a Maneuver Step .....	93
4.5.4	Global Settings / Preparation .....	95
4.5.5	Special Maneuvers - Definition of start and end conditions ..	98
4.5.6	Longitudinal Dynamics .....	100
4.5.7	Lateral Dynamics .....	103
4.5.8	Minimaneuver Command Language .....	107
4.6	Input From File / Using Real Measurements .....	110
4.7	Driver .....	116
4.7.1	Overview of IPGDriver .....	116
4.7.2	User parameterized Driver .....	118
4.7.3	Traffic .....	121
4.7.4	Racing Driver .....	122
4.7.5	Learning Procedure: Driver Adaption .....	123
4.7.6	Additional Parameters .....	125
4.8	Traffic .....	127
4.8.1	Overview of IPGTraffic .....	127

4.8.2	Defining Traffic Objects .....	128
4.8.3	Traffic Object Maneuver Description .....	134
4.8.4	Animated Pedestrian .....	138
4.8.5	Autonomous Driving .....	139
4.8.6	Output Quantities .....	140
4.8.7	Interaction with Driver Assistance Sensors .....	141
4.9	Environment .....	142
4.9.1	General Parameters .....	142
4.9.2	Model Parameters .....	143
4.9.3	Additional Parameters .....	144
<b>5</b>	<b>Parameterization: Vehicle Model</b>	<b>145</b>
5.1	The Vehicle Data Set Generator .....	147
5.2	CarMaker Coordinate Systems .....	148
5.3	Bodies .....	150
5.3.1	Model Kinds .....	151
5.3.2	Mass and Inertia Definitions .....	151
5.3.3	Flexible Body Mode .....	153
5.3.4	Coordinate System Fr1 and FrD for bodies .....	154
5.3.5	Other Parameters .....	154
5.4	Loads and Trim Loads .....	155
5.5	Engine Mount / Body .....	156
5.6	Suspension: Spring .....	159
5.7	Suspension: Secondary Spring .....	161
5.8	Suspension: Damper .....	162
5.9	Suspension: Buffer .....	163
5.10	Suspension: Stabilizer .....	165
5.11	Suspension: Kinematics .....	166
5.11.1	Basic Concept .....	166
5.11.2	Kinematic Models Overview .....	167
5.11.3	Linear Kinematics Models .....	168
5.11.4	Non-Linear Kinematics Model .....	170
5.11.5	Multi Body Suspensions Overview .....	172
5.11.6	Additional information .....	177
5.11.7	MBS: McPherson .....	179
5.11.8	MBS: Four Link .....	183
5.11.9	MBS: Twist Beam .....	188

5.11.10 MBS: Double Wishbone .....	192
5.12 Suspension Compliance .....	197
5.13 Suspension: Wheel Bearing Friction .....	200
5.14 Suspension: External Forces .....	201
5.15 Steering System .....	202
5.15.1 Steering Models Overview .....	202
5.15.2 Static Steer Ratio Model .....	203
5.15.3 Dynamic Steer Ratio Model .....	203
5.15.4 Pfeffer Power Steering .....	205
5.15.5 Truck Power Steering .....	212
5.16 Brake .....	216
5.16.1 Brake Control (Hydraulic System) .....	216
5.16.2 Brake System: Hydraulic System .....	218
5.17 Powertrain Overview .....	221
5.18 Powertrain: Drive Sources .....	224
5.18.1 Parametrizing the Engine .....	224
5.18.2 Parametrizing the Starter Motor .....	233
5.18.3 Parametrizing the Electric Motor .....	235
5.18.4 Parametrizing the Gearbox .....	239
5.18.5 Parametrizing the Clutch .....	242
5.18.6 Parametrizing the Planetary Gear .....	247
5.19 Powertrain: Driveline .....	248
5.19.1 Parameterizing the Differential Coupling Model .....	254
5.20 Powertrain: Control Units .....	257
5.20.1 Parametrizing PT Control .....	257
5.20.2 Parametrizing the ECU .....	267
5.20.3 Parametrizing the TCU .....	268
5.20.4 Parametrizing the MCU .....	271
5.20.5 Parametrizing the BCU .....	273
5.21 Powertrain: Power Supply .....	275
5.21.1 Parametrizing the Power Supply .....	275
5.21.2 Parametrizing the batteries .....	277
5.22 Aerodynamics .....	280
5.23 Sensors .....	282
5.23.1 Side Slip Angle Sensors .....	282
5.23.2 Body Sensors .....	283
5.23.3 Driver Assistance Sensors .....	285
5.23.4 Free Space Sensor .....	287

5.23.5	Traffic Sign Sensor .....	288
5.23.6	Line Sensor .....	290
5.23.7	Road Property Sensor .....	292
5.23.8	Satellite System .....	294
5.23.9	Collision Detection Sensor .....	296
5.24	Vehicle Control .....	298
5.25	Miscellaneous .....	299
5.25.1	Vehicle Graphics .....	299
5.25.2	Movie Geometry .....	300
5.25.3	Others .....	301
5.26	Trailer Model .....	302
5.26.1	CarMaker Coordinate Systems .....	303
5.26.2	Bodies .....	304
5.26.3	Loads and Trim Loads .....	306
5.26.4	Suspension .....	307
5.26.5	Brake .....	322
5.26.6	Hitch .....	324
5.26.7	Aerodynamics .....	325
5.26.8	Sensors .....	327
5.26.9	Miscellaneous .....	332
5.27	Tire Model .....	333
5.27.1	Selecting a Tire .....	333
5.27.2	Overview of Tire Models .....	335
5.27.3	Tire Data Set Editor .....	336
5.27.4	IPGTire .....	343
5.27.5	Pacejka / MF Tire .....	348
5.27.6	Magic Formula 6.2 by TNO .....	350
5.27.7	TameTire .....	350
5.27.8	Tire Data Set Generator .....	350
<b>6</b>	<b>Simulation</b> .....	<b>353</b>
6.1	Starting a TestRun .....	353
6.2	Information about Executable / Library .....	354
6.2.1	Start and Stop the Executable / Library .....	354
6.2.2	Information about the Executable and GUI .....	356
6.3	DVA: Online Manipulation of the Simulation .....	357
6.3.1	Usage .....	357
6.3.2	Application examples .....	359
6.4	Model Check .....	360

6.4.3	Overview of the Model Check . . . . .	360
6.4.4	Checking the Aerodynamics . . . . .	363
6.4.5	Checking the Powertrain . . . . .	364
6.4.6	Checking the Tire . . . . .	365
6.4.7	Checking the Brakes . . . . .	366
6.4.8	Checking the Driver . . . . .	367
6.4.9	Checking the Forces Distribution in the Suspensions . . . . .	368
6.4.10	Checking the Kinematics & Compliance . . . . .	369
6.4.11	Checking the design and static position of the vehicle . . . . .	378
6.5	Saving Results . . . . .	382
6.5.1	Starting the Saving Process . . . . .	382
6.5.2	Configuring the Saving Process . . . . .	383
6.6	Postprocessing . . . . .	386
6.6.1	Overview . . . . .	386
6.6.2	Postprocessing of the results using IPGControl . . . . .	386
6.6.3	Postprocessing using CONCERTO . . . . .	388
6.6.4	Postprocessing with MATLAB . . . . .	391
6.6.5	Data Conversion (ASCII, CSV, XLS) . . . . .	393
6.6.6	Converting Tables with resutil . . . . .	394
6.7	CarMaker Archive Service . . . . .	395
<b>7</b>	<b>Animation with IPGMovie</b>	<b>397</b>
7.1	User Interface Basics . . . . .	398
7.1.1	Navigating with the camera . . . . .	398
7.1.2	Display options . . . . .	398
7.2	Advanced features . . . . .	403
7.2.1	Simulation control with IPGMovie . . . . .	403
7.2.2	Replay . . . . .	403
7.2.3	Comparing two simulations . . . . .	405
7.2.4	Loading simulation data . . . . .	407
7.2.5	Quality Settings . . . . .	408
7.2.6	Advanced Camera Settings . . . . .	410
7.2.7	Camera lens types . . . . .	411
7.2.8	Multiple Views with IPGMovie . . . . .	415
7.2.9	Exporting videos and images . . . . .	417
7.2.10	Connect to Google Earth . . . . .	418
7.2.11	Integrating Dynamic Objects . . . . .	419
7.2.12	Integrating Light Assistance Systems . . . . .	420
7.3	Fine-tuning the animation . . . . .	421

7.3.1	Further Settings .....	421
7.3.2	The IPGMovie Interface in IPGRoad .....	423
7.3.3	Using Alternative Textures for Tree Strips .....	424
7.3.4	Displaying a User Defined Environment .....	426
7.4	IPGMovie Object Files .....	428
7.4.1	General Information .....	428
7.4.2	Integrating mtex-Files .....	431
7.5	Active Lights .....	432
7.5.1	Introduction .....	432
7.5.2	General information .....	432
7.5.3	Active Lights on the Road .....	434
7.5.4	Active Lights on the Test Vehicle .....	435
7.5.5	Control the Active Lights .....	441
7.5.6	Active Lights on Traffic Objects .....	442
7.5.7	Using Active Lights .....	444
7.6	Pedestrians .....	446
7.6.1	Basic technical information about animated characters in general ..	446
7.6.2	Available characters .....	447
7.6.3	Usage as traffic object .....	447
7.6.4	File types and folder structure .....	448
7.6.5	Pedestrian configurator .....	449
7.7	Video Data Stream .....	450
7.7.1	VDS Preview .....	454
7.7.2	Configuration File <i>VDS.cfg</i> .....	454
7.7.3	Streaming data over TCP/IP .....	458
7.7.4	Video Data Stream Client .....	459
7.7.5	Export image data to file .....	461
7.7.6	Performance considerations .....	463
<b>8</b>	<b>Test Automation</b>	<b>465</b>
8.1	Test Manager .....	466
8.1.1	The Test Manager dialog .....	467
8.1.2	Creating a Test Series .....	468
8.1.3	Result Quantification .....	478
8.1.4	CarMaker Test Configurator .....	480
8.1.5	CarMaker Test Report .....	483
8.2	ScriptControl .....	487
8.3	Variable Types .....	488

8.3.1	Named Values .....	488
8.3.2	Key Values .....	491
8.3.3	TestSpace Variables .....	494
8.3.4	ScriptFile .....	495
8.4	CarMaker Remote Control .....	498
<b>9</b>	<b>Miscellaneous</b>	<b>499</b>
9.1	Instruments Panel .....	499
9.2	Data Set Management .....	501
9.2.1	File Formats .....	501
9.2.2	Infofile Syntax .....	501
9.2.3	FileIdent .....	501
9.3	Data Encryption .....	501
9.3.1	Description .....	502
9.3.2	Kind of protection .....	503
9.4	CarMaker GUI Settings .....	504
9.5	Documentation / Help .....	506
9.5.1	CarMaker Bug Report .....	507
9.6	Simulation Parameters .....	509
9.7	TestRun Parameters .....	510
9.8	Session Log .....	510
9.8.1	Test Stand Usage .....	511
<b>10</b>	<b>Connecting to Third Party Tools</b>	<b>512</b>
10.1	Overview .....	512
10.2	ADAS: ADTF Interface .....	513
10.2.1	Overview .....	513
10.2.2	Configuration GUI .....	514
10.2.3	General Settings .....	515
10.2.4	Stream Settings .....	518
10.2.5	Structure of the Configuration Infofile .....	521
10.2.6	Example Configuration .....	524
10.2.7	C-Code API .....	526
10.3	Powertrain: CarMaker - CRUISE Interface .....	529
10.3.1	Setting up the CarMaker - CRUISE Interface .....	529
10.3.2	Simulating with a CRUISE Powertrain .....	529
10.3.3	Preparing the CRUISE Powertrain Model .....	531
10.3.4	CM-Car Interface IO-Signals .....	534
10.4	Powertrain: CarMaker - GT-SUITE Interface .....	539

10.4.1	Setting up the CarMaker - GT-SUITE Interface . . . . .	539
10.4.2	Simulating with a GT-SUITE Powertrain . . . . .	540
10.4.3	Preparing the GT-SUITE Powertrain Model . . . . .	541
10.4.4	VehicleCarMaker Interface IO-Signals . . . . .	543
10.5	Road: CarMaker - ADAS RP Interface . . . . .	550
10.5.1	Setting up the CarMaker - ADASRP Interface . . . . .	550
10.5.2	Export Routes from ADAS RP to CarMaker . . . . .	552
10.5.3	Using the Online Data Exchange between ADAS RP and CarMaker	
	556	
<b>A</b>	<b>FailSafeTester with CarMaker/HIL</b>	<b>558</b>
A.1	How it Works . . . . .	560
A.1.1	Wiring Topology . . . . .	560
A.1.2	Inside the FailSafeTester . . . . .	563
A.2	FailSafeTester Cards . . . . .	568
A.3	FailSafeTester Configuration GUI . . . . .	569
A.4	Manual FailSafeTester Configuration . . . . .	572
A.4.1	Card Naming Convention . . . . .	572
A.4.2	Configuration Settings . . . . .	572
A.5	FailSafeTester C-Interface . . . . .	577
A.5.1	Global Variables . . . . .	577
A.5.2	Configure the M51 CAN module . . . . .	577
A.6	Usage . . . . .	578
A.6.1	FailSafeTester Dialog . . . . .	578
A.6.2	FailSafeTester Mini-Maneuvers Commands . . . . .	586
<b>B</b>	<b>Minimaneuver Command Language</b>	<b>590</b>
B.1	Command Reference . . . . .	590
B.1.1	Driving Maneuver Commands . . . . .	590
B.1.2	Direct Variable Access Commands . . . . .	592
B.1.3	Extended RealtimeExpressions Commands . . . . .	595
B.1.4	Action Commands . . . . .	595
B.1.5	Logging Commands . . . . .	597
B.1.6	Data Storage Commands . . . . .	598
B.1.7	FailSafeTester Commands . . . . .	599
B.1.8	Resistors - Naming conventions . . . . .	601
B.1.9	Example of FailSafeTester Commands . . . . .	601

<b>C Realtime Expressions</b>	<b>602</b>
C.1 Fields of Application .....	602
C.2 Operators and Functions .....	603
C.2.1 Arithmetic Operators .....	603
C.2.2 Built-in Functions .....	604
C.2.3 Aliases .....	606
C.3 Application Examples .....	607
C.3.1 Realtime Expressions as Trigger Condition .....	607
C.3.2 Maneuver Definition .....	607
C.3.3 Trigger Minimaneuver Commands .....	608
C.3.4 ScriptControl programming language .....	608
C.3.5 Realtime Expressions in the Minimaneuver Command Language ..	609
C.3.6 Overwrite Quantities .....	609
C.3.7 Definition of new Quantities .....	609
C.3.8 Definition of new Variables .....	609
C.3.9 Syntax Examples .....	610
<b>D Model Check Quantity List</b>	<b>616</b>

## Chapter 1

# About this User's Guide

This document intends to deepen your knowledge about CarMaker you have already gathered reading the Quick Start Guide. The User's Guide intends to explain how to use each of the functionalities of CarMaker that are accessible through the Graphical User Interface (GUI). It should give you an overview of the options and features CarMaker offers. The focus is put on the explanations of functionalities and their application in the real vehicle development and testing process.

This User's Guide is held more application-specific and is a rich source of information about the handling of CarMaker. Contrary to the Quick Start Guide, it is not a tutorial. The technical background of the models used in CarMaker and the various interfaces should be looked up at the Reference Manual.

Everything that is explained in the Quick Start Guide is assumed to be known and serves as basic CarMaker knowledge. You should have already mastered the Quick Start Guide before you read this User's Guide and start to work with CarMaker effectively. In particular, before really starting to work with CarMaker, you should have understood the meaning of a TestRun, a model and a data set.

If so, enjoy this document and immerse in the large world of CarMaker with all its opportunities.

## Chapter 2

# General Overview

Before starting to simulate with CarMaker, we should explain the general structure of CarMaker, the meaning of the virtual vehicle environment (VVE) and the CarMaker Interface Toolbox (CIT).

This chapter is for information only and describes the whole possibilities CarMaker offers.

According to the product you bought (CarMaker/Office or CarMaker/HIL), you are able to do office simulation with CarMaker/Office or realtime simulation on a HIL test bench with CarMaker/HIL.

## 2.1 System Description

### 2.1.1 The Virtual Vehicle Environment - VVE

A *virtual vehicle* is a computer modeled representation of an actual vehicle with a behavior that matches that of its real world counterpart. With CarMaker, the virtual vehicle is made up of mathematical models that contain the equations of motion, kinematics, etc. along with other mathematical formulas that define the multibody system.

The model is then parameterized with data that relates directly to the vehicle to be studied. By this approach it is possible to use CarMaker to test any vehicle with a validated parameter set, and to easily switch between virtual vehicles by changing the parameter data that is used in the vehicle model. The virtual vehicle contains all parts of a real vehicle, including powertrain, tires, chassis, brakes, etc. It is also easy to integrate real automotive controllers (e.g. ABS, ESP, ACC, ...) or software modeled controllers into the virtual vehicle by using hardware or software in the loop.

A *virtual road* is a digitized or computer modeled representation of a road, track or course which simulates a real course or one that is generated specifically for testing. With CarMaker, the road can be generated in either of two ways:

- by combining individual road segments, such as straights and curves, to form a larger road. For each road section defined there is a specific length, width, angle, slope, pitch, friction coefficient, etc. that can be specified. Friction stripes, road shoulders, split mu conditions, wind machines, road markers and obstacles can also be configured.

- by using digitized data of an existing road that has been collected. With this approach, the measured data that is taken by survey or some other method is contained in a data file that is read by CarMaker and used as the road or test track during simulation.

A *virtual driver* is a computer driver, which simulates the actions of a real driver. Everything that would normally be controlled by a real driver, such as turning the steering wheel, stepping on the gas, brake and clutch, shifting gears in a manual transmission vehicle etc. is controlled by the virtual driver. There are two approaches that can be taken to perform the driver actions:

- simple control - with simple control, the actions are performed by specifying in advance what will happen at a certain time or distance. For example, you could say that at Time=N0 fully depress the clutch, at Time=N1 depress the gas halfway, at Distance=X depress the brake ninety percent and turn the steering wheel ten degrees to the left..
- IPGDriver - with IPGDriver, the course is controlled by a smart computer driver, which tries to maintain a course on a test track and can also plan ahead. The driver can be modified to operate within specified limits. For example, a driver may have a fast or slow reaction time, he may want to maintain a certain speed, he could be trying to go as fast as possible, etc.<sup>1</sup>

When the virtual vehicle, virtual road, and virtual driver are taken together, we can refer to it as the *virtual vehicle environment* (VVE), since the virtual vehicle “drives” on the virtual road and is “driven” by the virtual driver. [Figure 2.1](#) shows a graphical representation of the VVE.

### The Virtual Vehicle Environment

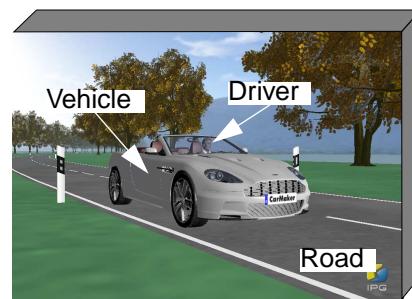


Figure 2.1: The VVE

## 2.1.2 The CarMaker Interface Toolbox - CIT

The second part of the CarMaker system includes all the tools that are used to manage the VVE. These tools will do things like: start and stop a simulation, select vehicle parameter data, define a vehicle maneuver, display results, show the progress graphically or as an animation, send and receive messages from the VVE, etc. We can call these tools the Car-Maker Interface ToolBox (CIT). Figure 2.2 shows the CIT in relation to the VVE.

As you can see from the figure, the CIT can be viewed as a number of individual tools that are used to manage the VVE. The tools can be classified as:

- *Control and Direct Access Tools* - control the actions performed by the simulation (e.g. start, stop, etc.) and also allow certain parts of the simulation to be directly controlled by the user (e.g. direct variable access, signal faulting).
- *Parameterization Tools* - serve to specify the parameters that will be used in the VVE.
- *Analysis and Visualization Tools* - allow the data to be viewed and analyzed either during or after a simulation (e.g. creating animations, plotting output quantities, etc.).

---

1. IPG-Driver is a very powerful tool, and therefore an additional document (the IPG-Driver manual) has been created to explain those features in detail, that are not or only briefly described in the *CarMaker User's Guide*.

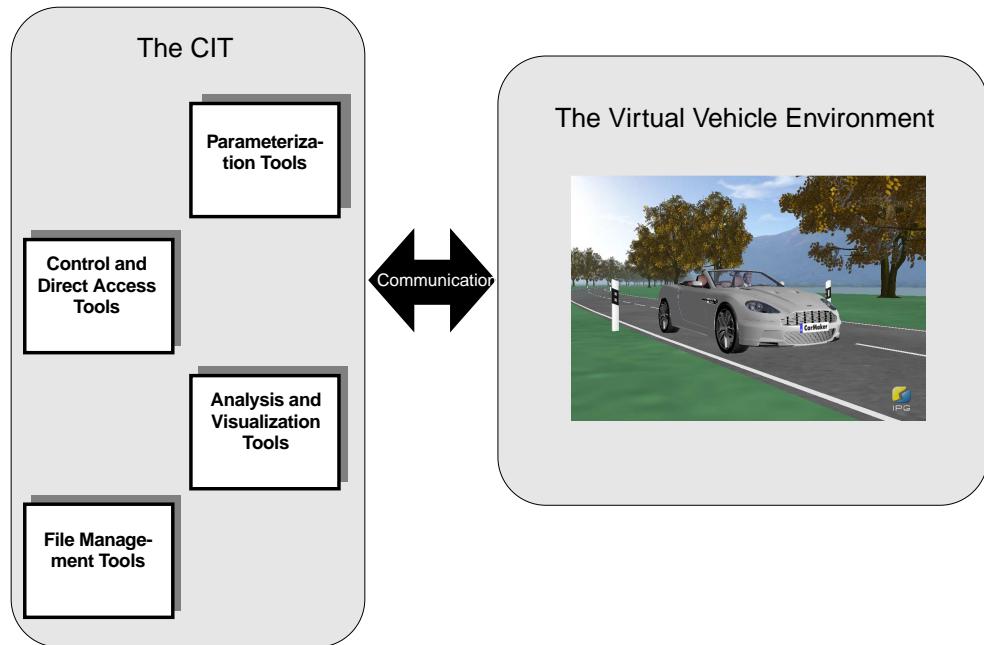


Figure 2.2: The CIT and the VVE

- *File Management Tools* - create, save and modify files that define simulations, settings, output data, etc.

Everything in CarMaker can be looked at in terms of the Virtual Vehicle Environment (VVE) and the CarMaker Interface ToolBox (CIT). However, this is only a conceptual model which is meant to present the idea behind to the reader in a clear way. In the next section we will discuss the CarMaker system as it is realized in hardware and software.

### 2.1.3 Real-Time vs. Office

Since different objects of investigation are examined when using CarMaker, there are also different approaches taken when implementing the VVE. The two approaches are:

- *realtime* - with a realtime approach the VVE is simulated on a computer that is running a real-time operating system. The real-time operating system enables the execution of the VVE to be deterministic, and the VVE will run on a timeline that corresponds to the timeline of the real world. If the VVE is not able to meet the criteria for real-time simulation, or in other words, fails to meet the specified timing deadlines, an error or warning will occur.
- *office* - with an office approach the VVE is simulated on an usual computer. There is no way to ensure a deterministic behavior, since the running VVE can be blocked or otherwise delayed by applications that run with a higher priority, or for some other reason depending on the way the operating system scheduling has been implemented. With an office simulation, the VVE may run with a timeline that is faster, slower, or equal to the real world timeline.

### 2.1.4 CarMaker/Office

When the object of investigation is a vehicle subsystem or a software modeled controller (as described in [section 2.2.1 'ECU Testing'](#)) and additional hardware does not need to be integrated into the VVE, the office approach (as described in [section 2.1.3](#)) should be used.

This configuration is called CarMaker/Office. [Figure 2.3](#) shows the CIT and VVE for CarMaker/Office.

In CarMaker both the CIT and VVE are running on the host computer<sup>1</sup> (Windows PC, Linux PC). No additional hardware is necessary and therefore is not shown in the diagram.

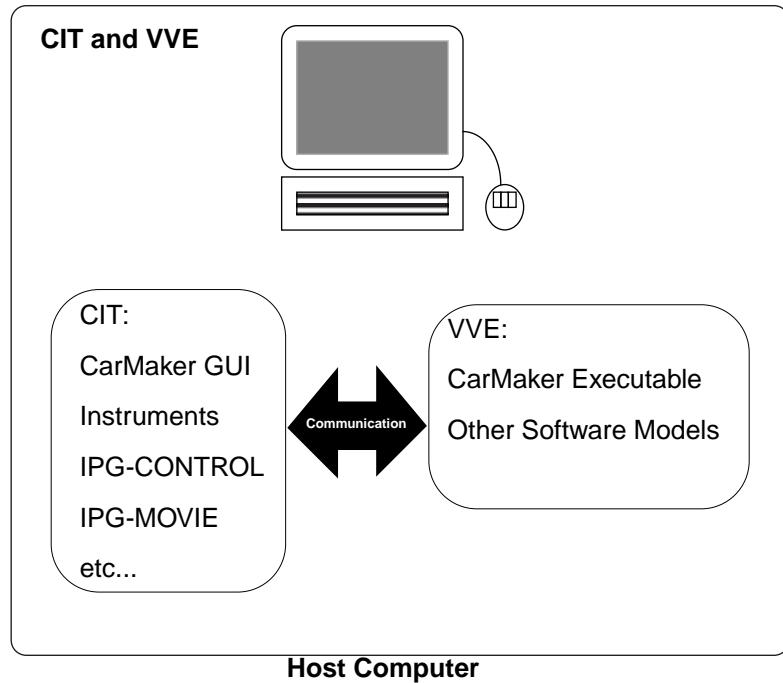


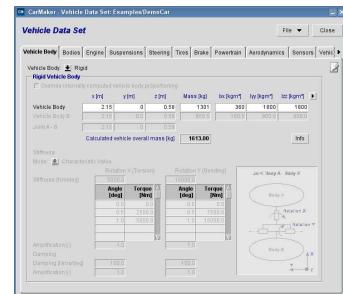
Figure 2.3: CarMaker/Office

## The CIT for CarMaker/Office

The CIT consists of a number of tools (applications and utilities) that run on a host computer. The host computer can be a Windows PC or Linux PC. The CIT includes the following:

### The CarMaker GUI

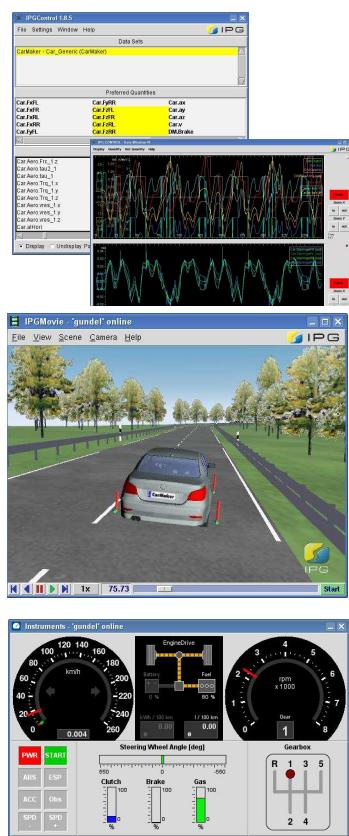
This is the main graphical user interface which is used to control the actions of the VVE, select the virtual vehicle parameter data, define or select the virtual road, set the virtual driver parameters, define or load maneuvers, open other tools that are part of the CIT, and a number of other useful operations.



### Vehicle Data Set Editor

Edit any parameter of the vehicle with a convenient user interface. Each individual submodel class is editable in a separate tab.

1. It is not necessary for the CIT and VVE to be run on the same host computer. If, for example, two networked computers have CarMaker installed it is possible to run the VVE on one machine, and the CIT on the other (both started in the same network mapped/mounted directory). Communication between the two will be done via TCP and UDP sockets, and with file I/O.



## IPGControl

Visualization and analysis tool. IPG-CONTROL can be used to view selected output quantities in real-time, load post-simulation data files, and plot and analyze the results.

## IPGMovie

Real-time 3D-animation of the VVE. The virtual vehicle is shown performing the specified driving maneuvers (performed by the virtual driver) on the virtual road.

## Instruments

Displays the most important instruments, dials and information about the vehicle's driving condition like: pedal position, steering wheel angle, gear selection, ignition, speedometer, tachometer, ESP- and ABS-warning lamps, brake light, etc.

## DVA

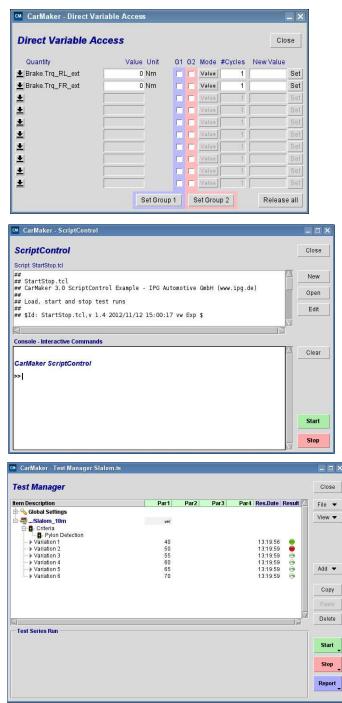
Direct Variable Access allows simulation quantities to be read and modified interactively through a user friendly graphical interface.

## ScriptControl

Test Automation utility that allows scripts to be defined, edited, and executed. All the functionality of the CIT can be controlled automatically using ScriptControl.

## TestManager

Another utility for test automation. A mixture of script and GUI based creation and execution of test series.



The simulation time can be controlled (either speed up or slowed down). Since real-time constraints have been removed, the simulation can be executed as fast or slow as desired (of course depending on the performance of the host computer that it is run on).

In addition to the tools described above, it is also possible to create custom tools that are programmed in either C/C++ or using Tcl/Tk. The custom tools can communicate with the VVE using a communications library (APO) that was designed by IPG. See the special APO documentation that is located in the folder doc of your installation directory, for more information.

## The VVE for CarMaker

The VVE for CarMaker/Office consists of the CarMaker Executable along with additional software modules (e.g. C/C++ or Simulink model) that are integrated into the VVE. Figure 2.9 shows the pieces of the CarMaker Executable.

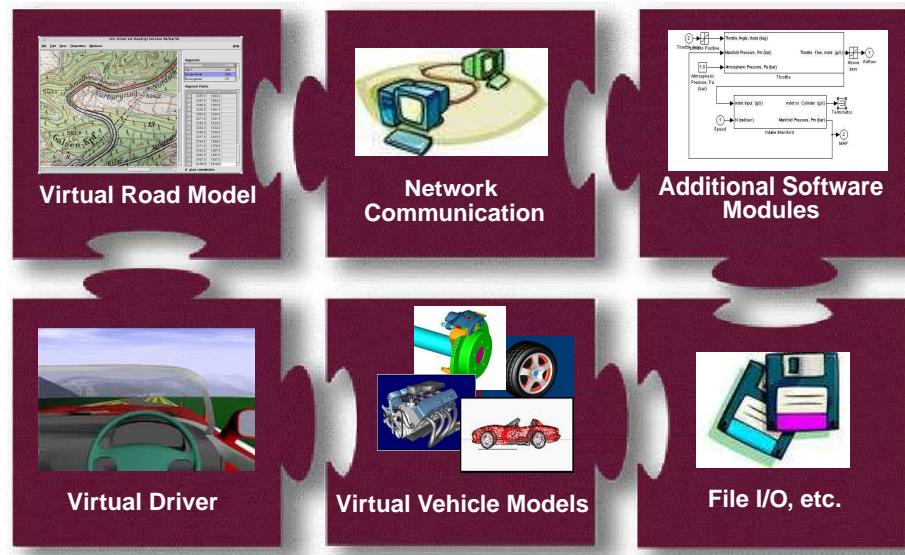


Figure 2.4: The Pieces of the CarMaker Executable

## Communication between the CIT and the VVE (CarMaker/Office)

The CIT and the VVE communicate in a number of ways, including:

- Network communication - messages are passed back and forth using the standard TCP/IP protocol. A special library designed by IPG (the APO library) is used to open TCP and UDP sockets and facilitate the communication between the CIT and the VVE running on the host computer.
- InfoFile database model parameter files (called InfoFiles in the CarMaker language), are shared between the CIT and the VVE. This is done by starting both the CIT and the VVE in one directory (the CarMaker working directory). In this way, files that are written and modified by the CIT can be read directly by the VVE, and alternatively the VVE can write files that can be read by the CIT. Synchronization is performed in the usual ways.
- Data storage - at any given time the VVE buffers specified data quantities and stores it in the working memory. When the user decides to save the buffered data, it will be stored to the hard disk, and can be analyzed with the tools provided in the CIT or with other applications, such as Matlab.

## 2.1.5 CarMaker/HIL

When the object of investigation is a hardware module or ECU, the real-time approach (as described in [section 2.1.3 'Real-Time vs. Office'](#)) is necessary. In the CarMaker system this configuration is called CarMaker/HIL. Which is written out in full: *CarMaker with Hardware in the loop*. [Figure 2.5](#) shows the hardware and software involved in a usual CarMaker/HIL configuration.

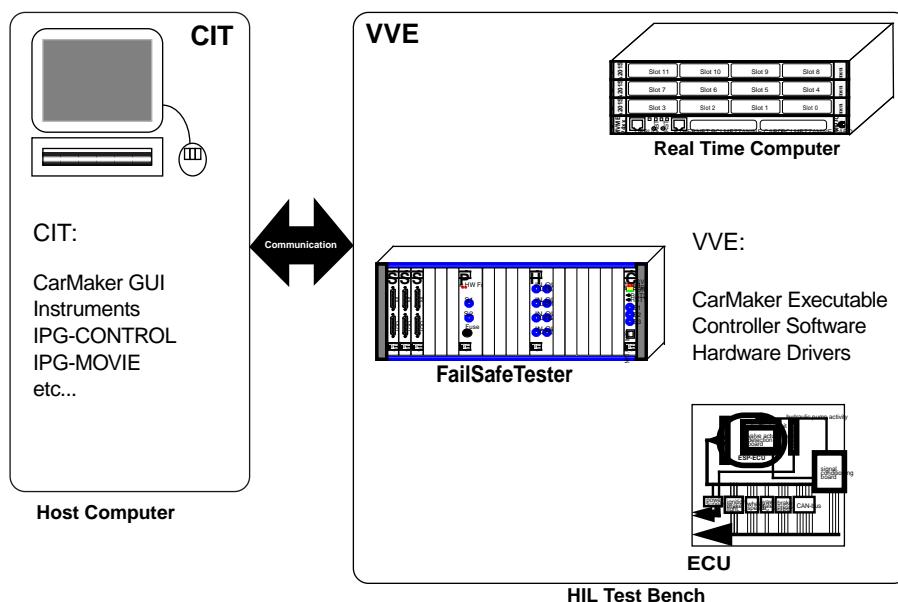
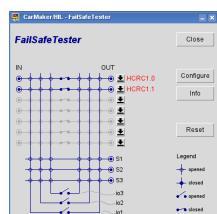


Figure 2.5: CarMaker/HIL

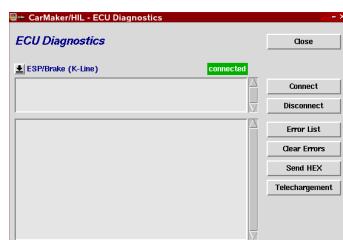
### The CIT for CarMaker/HIL

In the CarMaker/HIL configuration the CIT for CarMaker is the same as that for CarMaker/Office ([section 'The CIT for CarMaker/Office' on page 17](#)) with some additional tools:



#### FailSafeTester GUI

It controls the actions of the FailSafeTester which can generate electrical faults in the VVE.



#### Diagnostics GUI

Here you can send and receive information from an ECU using a K-Line or CAN protocol. Error codes can be read and hex commands can be sent to the ECU for implementation dependent things. This is a custom solution that is designed for a specific electronic control unit.

### The VVE for CarMaker/HIL

In the box labeled VVE in [Figure 2.5](#), three pieces of hardware are shown. In reality there can be much more hardware used to simulate the VVE, with hundreds of controllers and hardware modules potentially integrated into the hardware configuration. Also, the FailSafeTester is optional, but it can play such a vital role in ECU testing. Therefore, for the pur-

poses of this discussion we will call a VVE that includes one real-time computer, one test bench/ECU, and one FailSafeTester hardware unit, the standard CarMaker/HIL configuration.

### The Real-Time Computer

The real-time computer is the centerpiece of the VVE in the standard CarMaker/HIL configuration. The operating system for the real-time computer is LynxOS, which is a fully functional Unix based real-time operating system (RTOS). The used hardware is:

- Motherboard: MEN (Xeno), DSpace 1005 or 1006, Motorola MVME-Series
- Processor: Intel (Xeno), Power-PC (DSpace), AMD Opter (DSpace)
- Network: Fast Ethernet
- I/O modules: Analog I/O, Digital I/O, CAN bus, Frequency Generators, etc.

CarMaker/HIL uses the real-time computer to run the CarMaker executable, which is the real-time application that includes all the mathematical models for the virtual vehicle model and the virtual road model, the virtual driver functions, communication routines, I/O drivers, and other functionalities needed to link all the parts of the VVE together. [Figure 2.6](#) shows the parts of the CarMaker Executable that are run on the real-time computer. As you can see from the figure, the CarMaker Executable includes a number of features:

- Simulation of the road, vehicle and driver.
- Network communication (used to transfer messages to and from the CIT).
- Hardware I/O (e.g. data acquisition, CAN-bus, signal transmission) which is used to communicate to the external hardware and ECUs.
- File I/O - for logging data, storing results, reading parameter database files, etc.
- Additional functions - other internal software tasks necessary to simulate the VVE (e.g. system specific tasks, clean-up, etc.).

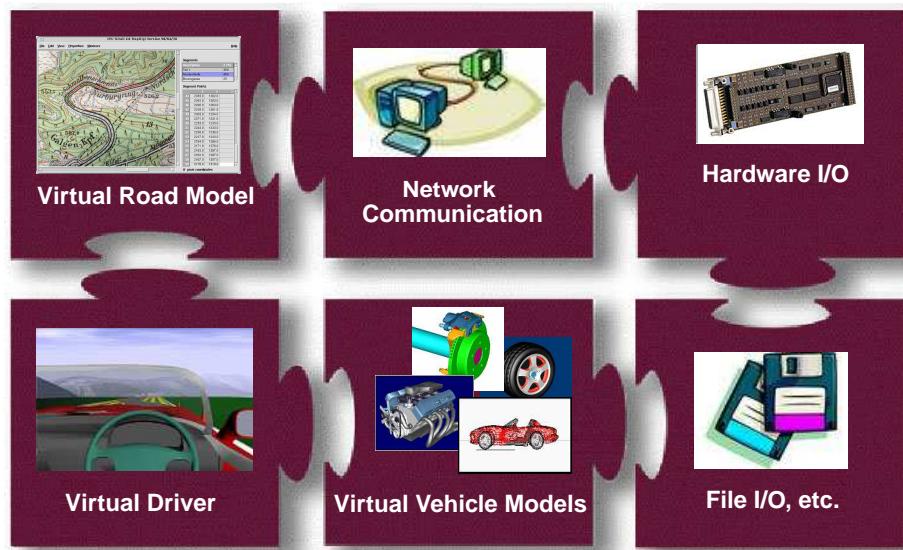


Figure 2.6: The Pieces of the CarMaker/HIL Executable

### Test Bench / ECU

In the context of this document, a *test bench* is made up of one or more hardware devices and/or ECUs that have been mounted on a board and wired appropriately to be used in the CarMaker/HIL system. It is then possible, but not necessary, to place the test bench in a

rack that could contain all the components of the VVE, including the real-time computer, FailSafeTester, or other hardware that is integrated into the VVE. [Figure 2.7](#) shows the basic schematic of a test bench that has an ESP mounted.

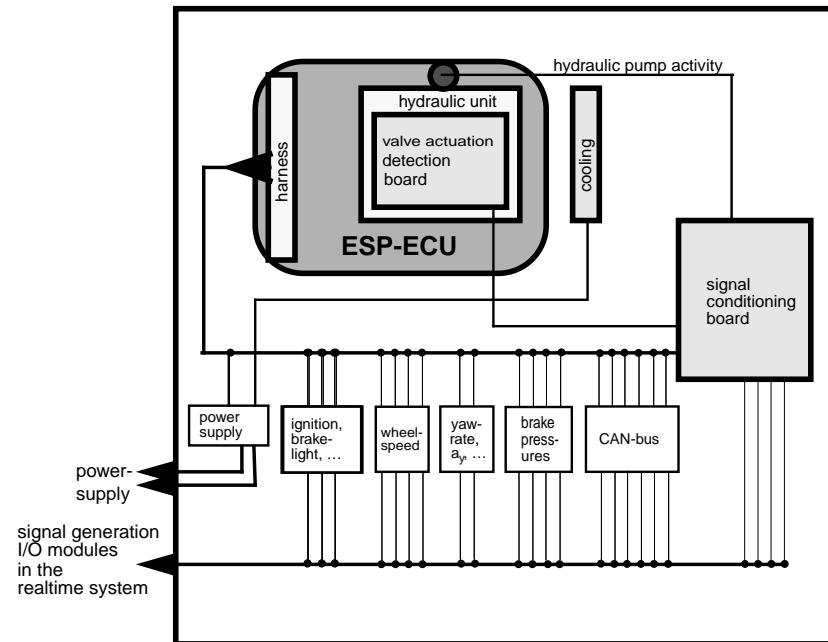


Figure 2.7: ESP Test bench Schematic

### FailSafeTester

The FailSafeTester is a hardware device that is used to generate fault conditions in the VVE's electrical environment. For example, signals that are run from the ECU to the real-time computer could be shorted, cut or modified otherwise, allowing the ECU to be performance tested for potential faults that could happen in the real world. Such conditions could be caused by wear and tear, improper installation or a number of other things that effect the electrical environment of the controller under study. Figure 2.7 shows a representation of the front of the FailSafeTester.

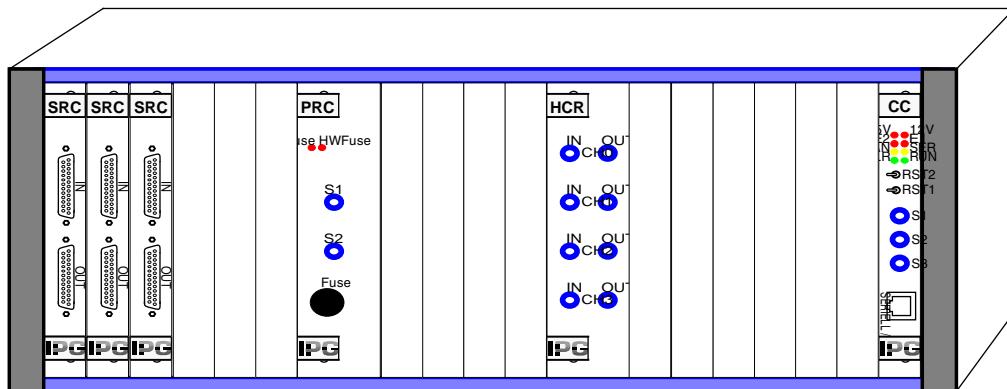


Figure 2.8: The FailSafeTester

## Communication between the CIT and the VVE (CarMaker/HIL)

The CIT and the VVE communicate in a number of ways, including:

- Network communication - messages are passed back and forth using the standard TCP/IP protocol. A special library designed by IPG (the APO library) is used to open TCP and UDP sockets and facilitate the communication between the CIT (running on the host computer) and the real-time computer (centerpiece of the VVE).
- InfoFile database model parameter files (called *InfoFiles* in the CarMaker vernacular), are shared between the CIT and the VVE. This is done by using NFS (Network File System) to allow the real-time computer to mount the file system on the host computer. In this way, files that are written and modified by the CIT can be read directly by the VVE's real-time computer, and alternatively the real-time computer can write files that can be read by the CIT. Synchronization is performed in the usual ways.
- Data storage - at any given time the VVE buffers specified data quantities and stores it in RAM. When the user decides to save the buffered data it will be stored to the hard disk of the host computer, and can be analyzed with the tools provided in the CIT or with other applications, such as Matlab.
- File sharing - additional files, such as files containing configuration data for the real-time system, are shared in the same way that the InfoFile database is shared.

### 2.1.6 From MIL and SIL to HIL

During the typical ECU development life-cycle it is frequently the case that a controller starts off as a software model before it is eventually realized with hardware. In the CarMaker System this is not a problem. There is a seamless continuity between the Office version that has a software modeled controller, and the real-time version with an actual ECU. You just have to make some changes to the VVE, like configuring a real-time computer with the necessary I/O and integrating the real controller in the system. Once this is done, all the tests that were performed with the office approach can then be implemented with the real-time approach, and no time is wasted in the transition from one to the other.

## 2.2 Fields of Application

As a product, CarMaker is normally used for several applications.

### 2.2.1 ECU Testing

- *Electronic Control Unit (ECU) testing* - to test components such as pumps, sensors, actuators, and their respective controllers by adding them to a virtual vehicle. In this way the hardware that would normally be installed in a real vehicle can be evaluated in a virtual vehicle.

This can be implemented with real ECUs which are physically connected to the system using I/O cards, CAN or some other methods. This way of testing is called Hardware in the Loop (HIL), since from the virtual vehicle's point of view actual hardware has been added to the system.

The other kind of ECU testing can use software modeled controllers that are added to the system by including tools like Simulink models, C-code, etc. in the virtual vehicle. This test procedure is called Software in the Loop (SIL), since from the perspective of the virtual vehicle there is a software element added to the system.

### 2.2.2 Subsystem Testing

- *Subsystem testing* - aims at testing the performance of a vehicle subsystem or how a subsystem effects the vehicle as a whole. For example, maybe the characteristics of the dampers have been modified in some way. Once the changes are made to the dampers of the virtual vehicle, a simulation can be run to determine how the changes effect the vehicle's handling.

There are other application in which CarMaker can be used, but these two cover the majority of applications.

## 2.3 Summary

The CarMaker system consists of two parts:

- the Virtual Vehicle Environment (VVE)
- the CarMaker Interface Toolbox (CIT).

The VVE simulates the vehicle, driver, and road (including wind, obstacles, traffic signs, etc.) which are all the parts that are required for evaluating a controller or testing the dynamics of a vehicle or vehicle subsystem.

The CIT allows complete control of the VVE, including direct interaction and control during a simulation, pre-simulation control definitions, model parameter database (InfoFile) editing, automated scripting and batch file creation, configuration changes and other functionalities. These manage all aspects of the VVE and, depending on the object of investigation, simulation can be performed using different approaches.

If, for example, an electronic control unit is being evaluated, then a realtime approach would be necessary. In other situations, such as subsystem testing, an office approach may be sufficient.



With the CarMaker System, the configuration of the VVE and CIT hardware and software depends on the object of investigation, but switching between the office approach and a realtime approach is straightforward. Therefore, it is easy to proceed along the development path from software based implementations to actual hardware prototypes of ECUs and hardware modules that are intended to be installed in a real vehicle once testing is complete.

For example, a controller could be modeled in software using Matlab-Simulink and integrated with CarMaker to test the behavior of the various algorithms. Then, once a real prototype controller becomes available, it can easily be integrated into the CarMaker/HIL environment, which allows the controller to be tested as if it was installed in a real vehicle. The controller performance can then be evaluated quickly and easily and optimized long before the real vehicle exists. Testing can continue as long as desired.

The rest of this document explains in detail how the CarMaker System should be used. The next chapter shows how to parameterize the virtual vehicle environment.

## Chapter 3

# Start CarMaker

Once you have completed the installation of the CarMaker software, received a valid license and saved it to the right folder, you can start working with CarMaker.

When CarMaker starts without any error messages, you completed the installation process successfully. Otherwise please have a look in the Installation Guide to verify the installations steps conducted, or contact the CarMaker Service Team

*CarMaker-Service @ipg-automotive.com*

for further instructions.

## 3.1 CarMaker/Office



To start CarMaker under Windows press the Start Button and select *Programs > IPG > CarMaker*.

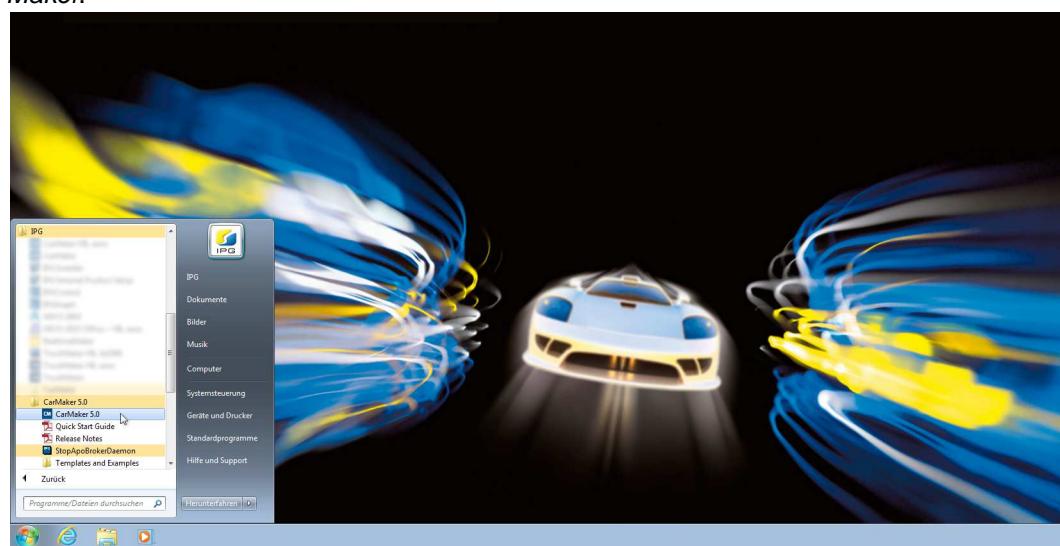


Figure 3.1: Starting CarMaker 3.0 and newer versions under Windows



To start CarMaker under Linux, open your console window like xterm or konsole. Type in the command

% CM &

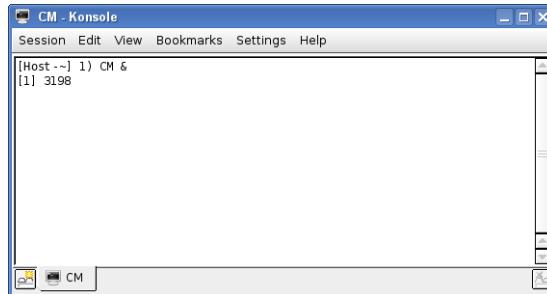


Figure 3.2: Starting CarMaker under Linux

The CarMaker main GUI pops up and loads automatically the project folder you were working in before you shut down the program at the last time. Nonetheless, you can easily switch the project folder by selecting *File > Project Folder* and select the one you like to work in.

## First Start

When you start CarMaker for the first time you need to create your first project directory. In the GUI, the only option available is under *File > Project Folder*. Select *Create / Update Project*.

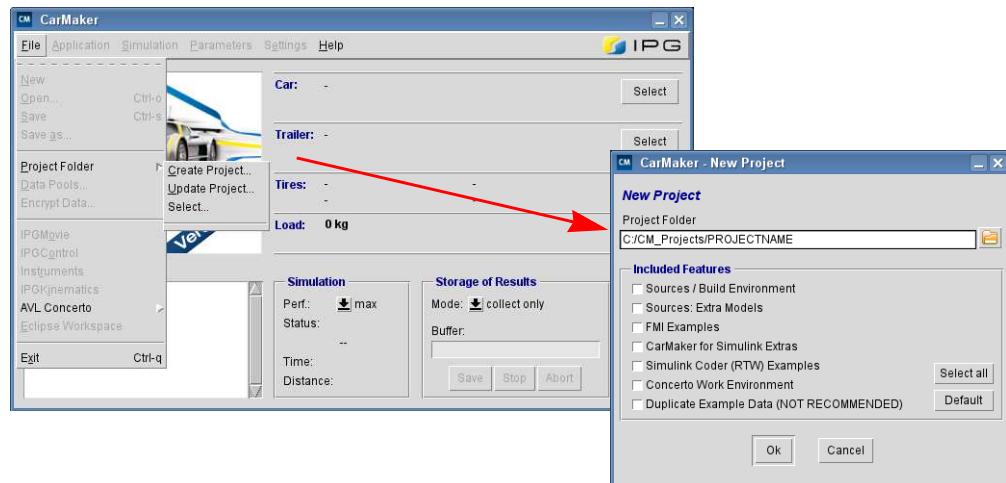


Figure 3.3: Creating a new project directory

In the popping up dialog box you need to enter a path where your project directory should be started. Select the features depending on the version you are using and confirm with ok.



Hint: Create a central folder called "CM\_Projects" and add every new project to this folder.

Depending on the application case, you can add several features to your project folder. For basic simulations, none of these features are required, though.

Table 3.1: List of project features

Feature	Description
Source / Build Environment	Required for compiling new CarMaker executables

Table 3.1: List of project features

Feature	Description
Sources: Extra Models	Adds several example models written in c-code to the src folder of your project
FMI Examples	Loads various example Functional Mockup Units to the Plugins folder of your project
CarMaker for Simulink Extras	The src_cm4sl folder will be extended by various Simulink models showing how to extend the Car-Maker environment
Simulink Coder (RTW) Examples	Provides model examples, that were integrated via Matlab Simulink Encoder (former RealTime Workshop)
Concerto Work Environment	If the post processing tool AVL Concerto is available on the system, predefined layouts from the CarMaker project folder can be used.
Duplicate Example Data	This option copies all example data files (TestRuns, vehicle and tire data sets, etc.) to your project folder. This feature is usually not required. Without this option, you still have access to all examples stored in the installation directory.

## 3.2 CarMaker for Simulink

To start CarMaker for Simulink a project folder needs also to be created.

Then, just open Matlab and navigate to the “src\_cm4sl” folder of your project directory and open one of the provided .mdl files (generic.mdl is the simplest running model).

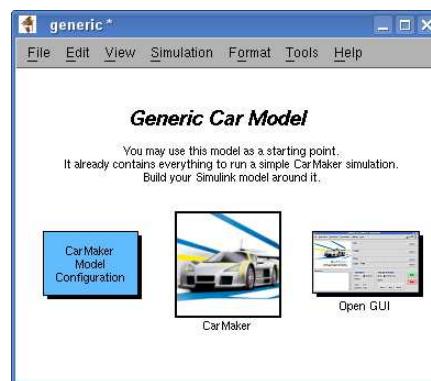


Figure 3.4: Top view of the generic.mdl in CarMaker for Simulink



Hint: If you are using MotorcycleMaker you may have to change the subdirectory to *src\_mm4sl* or to *src\_tm4sl* respectively if you are using TruckMaker.

With double clicking the OpenGUI block you will start the CarMaker GUI. If necessary to start it with optional starting parameters type into the Matlab Command Window:

```
CMDData.GuiArgs = { '-help', '-foo', '-bar'}
```

and if you want to start a GUI which is located somewhere else use following command:

```
CMDData.GuiExe = '<path to alternate gui>HIL.exe"
```

Up to here you can proceed similarly to CarMaker/Office.

More information about the usage and features of CarMaker for Simulink can be found in the Quick Start Guide, chapter “Simulating with CarMaker and Simulink” and in the Programmer’s Guide, section “CarMaker for Simulink”.

## 3.3 CarMaker/HIL



First of all you have to configure your realtime computer. Please see the “Installation Guide” for details.

### 3.3.1 Remote Start for CarMaker/HIL Xeno

To connect the CarMaker GUI with a Xeno realtime system and start the simulation, a dialog based remote start option for the realtime unit is available. The dialog can be found in the CarMaker main GUI under *Realtime System > Configuration / Status*.

The upper part of the window shows the path to your current CarMaker project directory.

**Command (executable)** Select the CarMaker.xeno or CarMaker.labcar executable that should be operated by the realtime system. A drop down menu shows commonly used executables for a fast selection.



Please note, that the CarMaker/GUI and the CarMaker application running on the realtime system need to be within the same project folder.

**Command line options** Especially in the HIL environment, command line option are frequently used, e.g. to choose between different IO configurations. The options start with a hyphen, e.g.

```
-io none
```

If more than one option should be used, use a blank as separator.

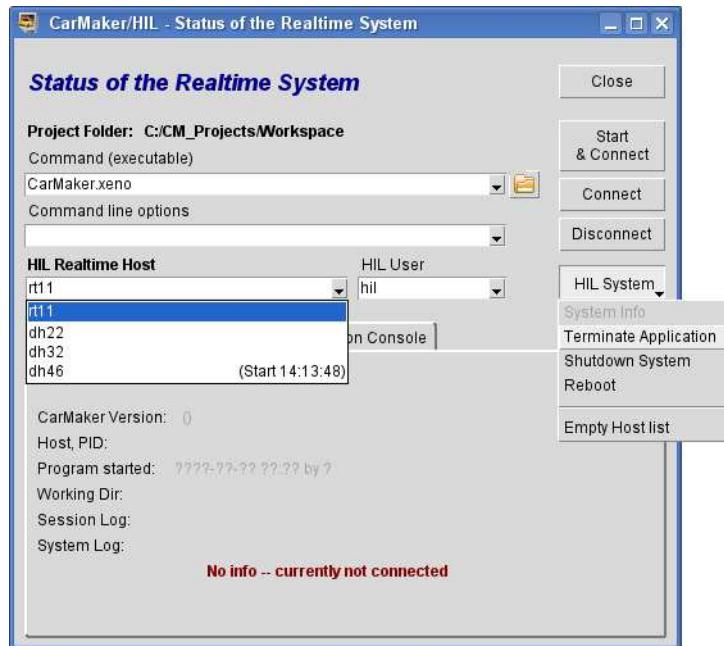


Figure 3.5: CarMaker/HIL remote start dialog

- HIL Realtime Host** The drop down menu lists all realtime units available. Select the target system from the list.
- HIL User** Specifies the log in at the realtime system. The user has to be defined in the realtime system configuration (see Installation Guide for details).
- Start& Connect** Establishes a connection with the target HIL system specified under *HIL Realtime Host* and starts directly the application.
- Connect** This button only connects the CarMaker GUI with the realtime system. The application (executable) is not started yet.
- Disconnect** Cuts the connection between the CarMaker GUI with the realtime system.
- HIL System** By holding this button, a drop down menu pops up. From here, the realtime system can be remote controlled:
- The running CarMaker application can be stopped (*Terminate Application*)
  - The whole realtime system can be turned off (*Shutdown System*)
  - The realtime system can be rebooted (*Reboot*)
- Once a connection between CarMaker GUI and realtime system has been established successfully, additional information about the running application and the compilation can be found in the lower section of the remote start dialog (tabs *Application Status* and *Compile Info*). The realtime unit's command line output is forwarded to the tab *Application Console*.

### 3.3.2 Manual Connecting to CarMaker/HIL Xeno

The second approach to start and connect with a realtime system is to establish a telnet session through the command line. This is the only start option for CarMaker/HIL versions older than 4.5.

When your realtime computer runs properly, just

- open a command window and log in your realtime unit using "telnet",

- navigate to the project directory using “cd”,
- start CarMaker from here using “bin/CarMaker.xeno”,

```
[Host-~] 1) telnet rt6
Trying...
Connected to rt6.
Escape character is '^'.

rt6 login:
Password: xx
hallo
[rt6-xx] 1) cd /fs/u3/CarMaker/Projects/src
[rt6-src] 2) CarMaker.xeno -screen
APPLICATION      Car_Generic <insert.your.version.no> #173 (3.0)
COMPILED         xx@Host.ipg.de
IO Configuration: 2
rt6 should be connected.

ESP rig: Calibration of Hall Sensors
=====
Calibrated Offset for Sensor 0: 2.462806
Calibrated Offset for Sensor 1: 2.495766
Calibrated Offset for Sensor 2: 2.455482
Calibrated Offset for Sensor 3: 2.470130
Calibrated Offset for Sensor 4: 2.498817
Calibrated Offset for Sensor 5: 2.464027
Calibrated Offset for Sensor 6: 2.462196
Calibrated Offset for Sensor 7: 2.490883
Calibrated Offset for Sensor 8: 2.479896
Calibrated Offset for Sensor 9: 2.501259
Calibrated Offset for Sensor 10: 2.466468
Calibrated Offset for Sensor 11: 2.460365

SIM START      Idle
TIME    0.000
```

- start the CarMaker GUI on your host machine.



To start CarMaker GUI under Windows press the Start Button > Programs > IPG > CarMaker/HIL.



To start CarMaker/GUI under Linux open a second window of your command line. Move to your project folder and start CarMaker/HIL.

```
% cd /local/CM_Projects/ABSTests
% CM_HIL -apphost <rtsys> -connect . &
```

The dot at the end of the command line is mandatory and lets the CarMaker GUI start from the project folder you are working in. Thus, you do not need to select the project folder at the GUI.

### 3.3.3 CarMaker on dSPACE Systems

There are two possible ways to run CarMaker HIL in cooperation with a dSPACE Real-Time System and to adapt it especially to your requirements.

One possibility is to use the c-code interface. This approach is described in the following step. If you would like to use Matlab Simulink and the real-time workshop, you can go on with the section *Building a new CarMaker executable using Matlab*.

#### Building a new CarMaker executable using c-code interface

The *Makefile* file is located in the *C:\CM\_Projects\<Project\_Name>\src* directory on your host PC for the building process with GNU make, which is accessible through development environments like MSYS.

Depending on your system configuration it may be necessary to comment out the makefile variable *DSPACE\_ROOT* pointing to the dSPACE installation directory. The same applies to the *PPC\_ROOT* variable if you are using a DS1005 compiler and respectively to the *X86\_ROOT* variable if you are using a DS1006 compiler.

## Building a new CarMaker executable using Matlab

If you are starting Matlab for the first time, a pop-up will appear after opening Matlab. Within this pop-up, you can select the supported dSPACE RTI platform you would like to use. At a later time, this can be changed again by typing in *rti<Board\_No>*, where <Board\_No> stands for the respective number of your real-time system.

To load a model, you have to choose the directory. To do this, you have to select the *<Project\_Name>\src\_cm4sl* path using the *Current directory* area at the top of the Matlab interface. After this folder is selected, double-click on the *generic.mdl* model located in the filebrowser on the left side of Matlab.

Before you can proceed a building process, you once have to select a system target file. Select *Tools > Real-Time Workshop > Options* and open up the *Real-Time Workshop* tab to select the system target file. Relating to your dSPACE RTI platform, this is *rti<Board\_No>..tlc*. If this is done, select *Tools > Real-Time Workshop > Build Model* to build and download your new CarMaker Real-Time Application.

The next step is to assign the new executable to your CarMaker environment, which is explained in the following step.

## Connecting CarMaker with the dSPACE real-time computer

To connect CarMaker to the dSPACE system using the customized executable, select *Real-time System > Configuration / Status* within the Carmaker GUI and select your CarMaker executable in the dropdown menu. To establish the connection, select *Start & Connect*.

Once the executable is defined, you just need to select *Realtime System > Start & Connect* to establish a connection of CarMaker to the dSPACE system.

### 3.3.4 CarMaker with ETAS Hardware

For information regarding the usage of CarMaker with ETAS hardware, have a look at the section *Using CarMaker/HIL with ETAS Hardware* in the Programmer's Guide.

## 3.4 Directory Structure

### 3.4.1 Project Directory

When you run a simulation in CarMaker, a lot of data is needed for calculations. The required data is saved to a certain file structure, so that CarMaker exactly knows every time where to find the data. This file structure has to be the same for every simulation environment and is called *project directory*. It contains all data required for a simulation, e.g. vehicle data sets, tire data, road and maneuver settings etc. Of course, the data itself can vary between different projects, but the way it is arranged must always be the same.

This project folder is the proper place to save all data relevant for your current project.

Please note: with CarMaker 5.0 onwards, the example data files are saved in the CarMaker installation directory, so that the project folder really only contains your data files.

Below you can find the structure of a typical project directory (the structure of CarMaker versions older than 3.0 can differ slightly):

Listing 3.1: Structure of a working directory

```
<project directory path>

    -bin           Executables, user specific
                  GUIs, like e. g. Instruments

    -Concerto      Working directory for postprocessing tool Concerto including layouts,
                  formulas, scripts etc.

    -Data          Data basis
        | -Chassis   Kinematics and Compliance of axles
        | -Config     Configuration of the test bed: ECUParameters, SimParameters,
                      OutputQuantities
        | -Misc       Misc., e.g. parameters for the hydraulic
        | -Pic        Views of vehicles (for GUI)
        | -Road       Measured road definitions (RoadData)
        | -Script     ScriptControl test scripts
        | -TestRun    Test runs
        | -Tire       Tire data
        | -Trailer    Trailer data
        | -UserDriver User defined driver model
        | -Vehicle    Vehicle data
        | -VehicleControl Vehicle data

    -doc           Online documentation

    -include        Header files for CarMaker C-interface

    -lib            Project specific CarMaker library

    -Movie          Animation, movie
                  Vehicle and road geometry files

    -Plugins        Storage place for Functional Mockup Interface Units (FMUs)

    -SimInput       Data for Input_From_File
    -SimOutput      Results of simulations
        | -Offline
        | -rt1
        |   | -Log
        |   | -YYYYMMDD

    -src            Development environment (user accessible source files)

    -src_cm4sl     Development environment for CarMaker for Simulink
```

The data files contained in the single directories are called *Infofiles*. They are plain ascii text files with a specific syntax, as CarMaker uses keyword oriented parameter files. Further information on the buildup and the keywords of these Infofiles can be found in the Reference Manual.

### 3.4.2 Installation Directory

At this point, it also might be useful to show the CarMaker installation directory, e.g. where the CarMaker tools have been installed, along with the CarMaker libraries and documentation. Below, an example of the installation directory of the Xeno (realtime) version of CarMaker, installed in the Unix environment is shown.

Listing 3.2: Structure of the installation directory

```
<installation directory path>

-ADASRP-IF          Files relevant for the interface to ADAS RP
-bin                 Programs like Instruments, tireutil, resutil
-bin-xeno            Interface for CarMaker/HIL on Xeno
-CM4SL              CarMaker for Simulink interface
-Data                Example data like TestRuns, vehicles, tires, etc.
-doc                 Online documentation
-Examples            Exemplary C-Code extensions for CarMaker
-GUI                 Graphical User Interface scripts, executables, etc.
-include              Include files
-lib/lib64            Libraries for the 32-bit and 64-bit installation
-Matlab               Support package for Matlab/Simulink
-MM4SL               MotorcylceMaker for Simulink interface
-Movie                Configuration files for IPGMovie
-Plugins              Configuration files for the FMU interface
-Setup                Template and definition files
-SimInput             Example profiles for InputFromFile
-Templates            Example projects
-TM4SL               TruckMaker for Simulink interface
-TrafficSigns         Library of traffic signs
-xeno_rt              Xeno runtime environment
```

### 3.4.3 Shared Directory

The file browser in CarMaker 4.5 has capacity of reading Central Data Directories (Data Pools) in addition to the project directory and installation directory. This is configured using Data Pools dialog available under *File > Data Pools*.

The Datapool should have the same folder structure as the project folder (<project folder>/Data/...). Otherwise, CarMaker will not recognize the corresponding files.

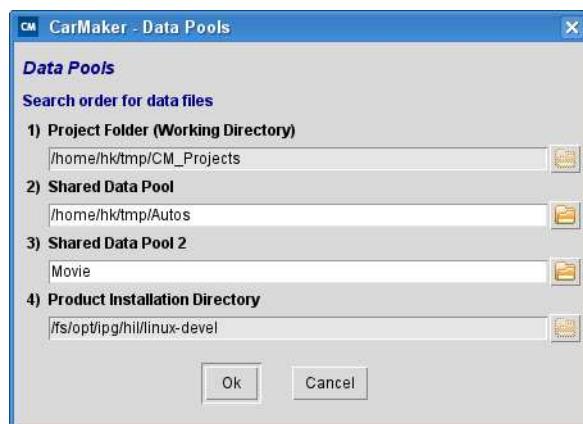


Figure 3.6: Dialog to specify shared Data pools

The file browser is very handy and can be used at various CarMaker interfaces for selecting parameters:

- TestRun
- Vehicle
- Tire
- Road
- Maneuverer
- Driver
- Input from File
- Environment
- Traffic
- Vehicle Graphics and Movie Object
- Suspension Parameters
- Brake Parameters
- Traffic



While using shared directory if two or more files happen to have same file name, Car-Maker will use the file from project folder during simulation and ignores the others.

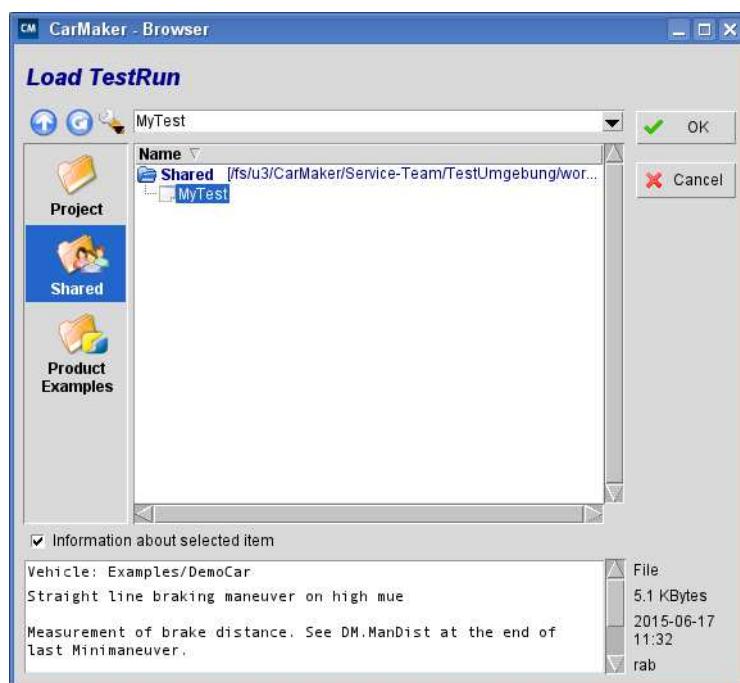


Figure 3.7: Example showing file selection from Shared Directory 1.

## Chapter 4

# Parameterization: TestRun

## 4.1 Definitions

A TestRun is a test scenario which collects all the information required to parameterize the virtual vehicle environment and to start a simulation. Depending on complexity of the simulated test case, the TestRun composes of a different number of modules. As minimum requirement to be able to simulate, the following modules have to be parameterized within the TestRun:

- **Vehicle:**  
Definition of the vehicle data set used.
- **Road:**  
Parameterization of the test track.
- **Maneuver:**  
Mainly to specify the driver's task.
- **Driver:**  
Set driver behavior (defensive, normal, aggressive, ...)

Additionally, the following modules can be defined in the TestRun, depending on the field of application:

- **Trailer:**  
To simulate a test car with trailer configuration.
- **Tires:**  
Overwrite the default tire data set referred in the vehicle model.
- **Traffic:**  
Add other static or moving traffic objects.
- **Environment:**  
Configuration of the test environment with date, time and ambient conditions.

All this information is stored within the TestRun. Some information is directly written to the TestRun *Infofile*, others are referenced by a link to a second file (see [section 4.2.2 'File System Overview'](#)).

Please find a detailed description of all modules in the following chapters.

Please Note: The TestRun parameters also define, which technical models are simulated. A model is a mathematical formulation of a mechanical or physical system. The models in CarMaker are already available, which means you do not need to wonder about how to develop a vehicle model. In particular, the vehicle, trailer, and kinematics models are MBS models (Multi Body System).

Do not mistake the CarMaker model for a "Simulink model". When it works with Simulink, CarMaker still uses the same models. But in that case, you can see how the various models which are part of CarMaker are linked to each other. The Simulink interface enables you to use the signals that flow between the CarMaker models, either to calculate another additional variable you want to use in a controller, or to modify them. In other words, the Simulink interface is a layer over CarMaker. In order to open this Simulink interface, you have to load a \*.mdl file: it is this \*.mdl file which is referred to as "Simulink model".

## 4.2 Data Management

### 4.2.1 Load and Save

In order to be able to simulate, CarMaker requires a TestRun to be loaded.

You can load an existing TestRun, save it, save it under a different name or create a brand new TestRun from scratch. You can access those functionalities through the menu "File" of the main GUI.

At least at the beginning, it is better to load an existing TestRun that is similar to the simulation you want to perform and to save it under a new name. There are various example TestRuns available in the *Product* folder on the left navigation menu.



Note that you are not obliged to save the TestRun: you can load / create one, parameterize it, and simulate without having it saved before.

Note that in the window that pops up when you want to load or save a TestRun under a new name, you can right click in it: a menu list appears with the most frequently used options that are usually available in a directory browser (new, delete, rename, attributes, show hidden files).

When you save a TestRun under a new name, you can add some comments in the field that is at the bottom of the window. These comments will then be displayed when you want to load this TestRun.

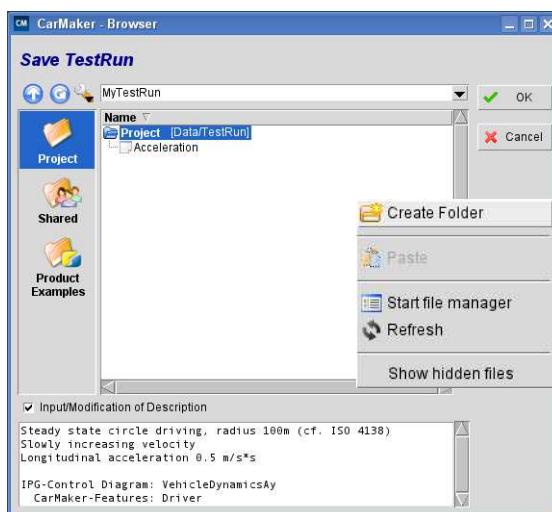
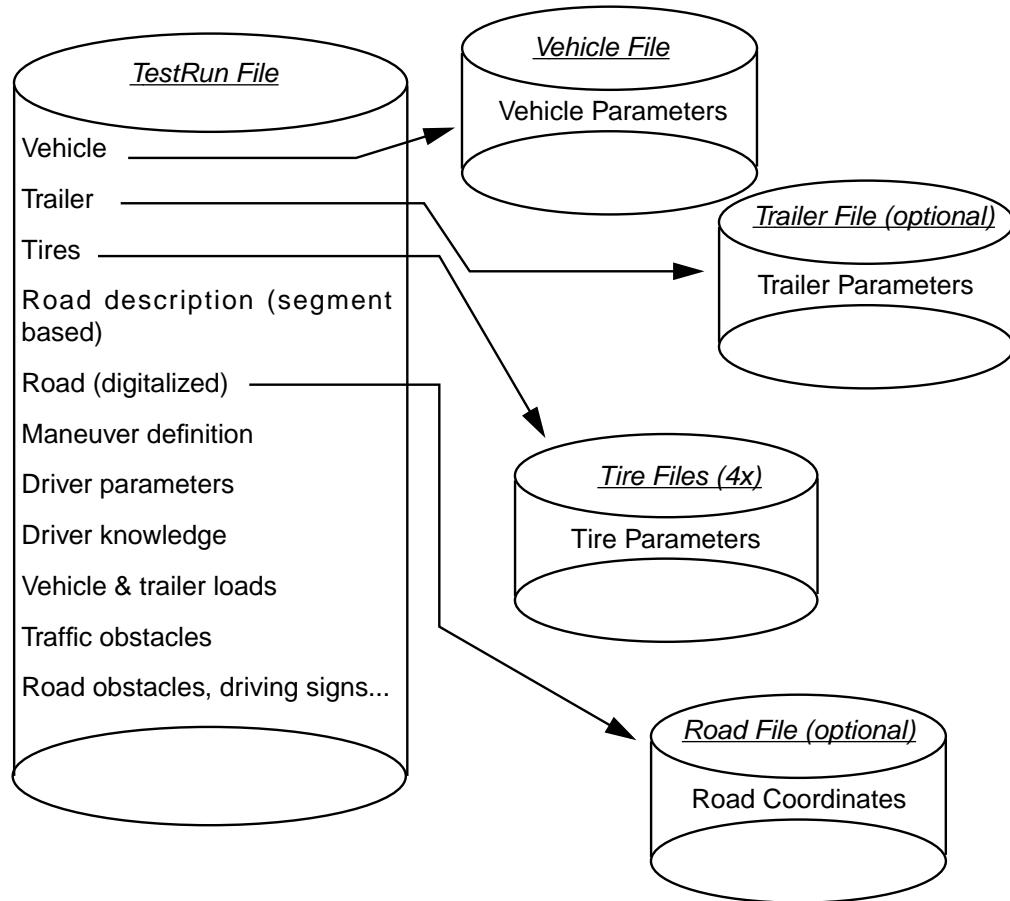


Figure 4.1: Renaming and saving a TestRun

## 4.2.2 File System Overview

The following picture illustrates which information are saved in the TestRun file, and which are saved in other files:



→ indicates a link to the file where the actual data is saved



As you see, if you save a TestRun, you will save the name of the vehicle used in this TestRun, but not directly the changes in the vehicle parameterization. Thus, do not forget to save the changes in the vehicle GUI, too. Anyway, if you have not done it before you close CarMaker, a message will warn you that there are unsaved changes in the vehicle dataset (see [section 5.1 'The Vehicle Editor' on page 145](#)). The same goes for the tires and trailer.

## 4.3 Starting from Scratch

In CarMaker the preparation for a vehicle test is very similar to how it would be done in the real world.

- Select a vehicle (if it does not exist, first build a vehicle data set).
- Choose or define a test track with obstacles and the desired conditions.
- Specify the type of driver or use a simple closed loop controller.
- Specify the tires and loads.
- Define the Maneuver(s).

- Optionally, select a trailer.
- Save the TestRun.



To learn more about how to build a TestRun, please read the Quick Start Guide, section “Building your TestRun from Scratch”.

The next sub-section explains how to configure the virtual environment of a TestRun: you will get a close insight in selecting and parameterizing new data sets. An overview of the CarMaker GUI shows you where to find the different sub-models to enter the data required to build a TestRun.

## 4.4 IPGRoad

The following sub-sections describe all parameters required to parameterize IPGRoad (accessible via the CarMaker GUI by clicking on Parameters > Road).

IPGRoad, the road model of CarMaker, enables to calculate the position of any point on the road surface, including its coefficient of friction, if desired.



Figure 4.2: View on the road in IPGMovie

### 4.4.1 Overview of IPGRoad

Using IPGRoad enables you to build open and closed tracks in CarMaker.

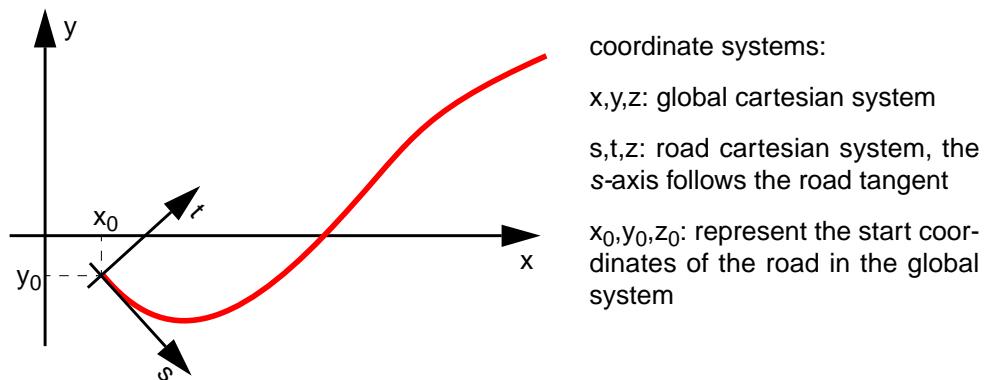
Basically, it describes the center line of the road ( $x, y, z$ ), which is symbolized by the white stripes on the picture above. Once the road geometry is defined, the track width counting from the centerline to both sides of the road has to be parameterized, too. The same goes for the friction coefficient of the road surface. In addition to that, you can redefine the width and friction coefficient only for a certain section of the road.

Furthermore, you can add obstacles and markers to the road, which can be used by other modules of CarMaker, e.g. velocity signs which will be taken into account by the driver model IPGDriver.

Once the TestRun is started, the road definition is automatically linked to IPGMovie.

In general, the road interface in CarMaker proposes two ways to parameterize the geometry of the road's center line, which will be explained in detail in the next two sub-sections:

- digitized real road measurements ( $x,y,z$ ),
- a virtual road with a segment-based interface.



## 4.4.2 General Settings

The IPGRoad dialog box can be accessed from the main GUI by selecting *Parameters > Road*. The first tab that appears shows the general settings of the road.

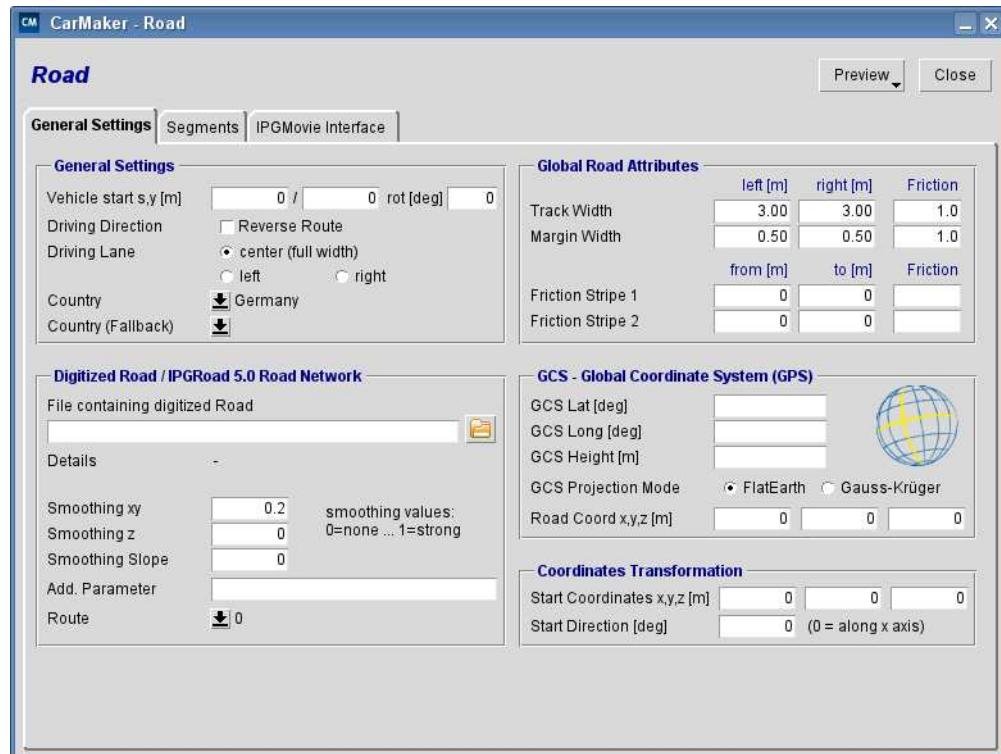


Figure 4.3: General Settings of IPGRoad

Using this tab, you can define the global settings which will be applied to the whole track.

**Vehicle start s,y (m)** Using this field, you can define the exact start position and orientation of the test car in longitudinal (s-coordinate along route centerline) and lateral coordinates and rotation angle respectively. The point of the car which is considered, is the origin of Frame1 (see *Reference Manual*, chapter 1.2 "CarMaker Axis Systems"). This feature is especially useful on a very long road, if you want to check the vehicle's behavior at a given point that is quite far, or to check the road geometry at this point.

**Driving Direction** This option will reverse the course, *Bumps* and *Markers* will not be rotated, however.

**Driving Lane** This option lets you choose which driving lane should be used: the left or the right one. With the option "center (full width)" the driver will be geared to the center line of the road and, depending on the corner cutting coefficient, will use the entire track width.

In the box "Global Road Attributes" the general layout of the track like its width and friction conditions can be parameterized. These settings will be applied to the entire road. If you want to overwrite some attributes for a certain track segment, refer to the description of section '[Override Selected Segment Attributes](#)' on page 47.

**Country** Right now this option only affects the visualisation and availability of the traffic signs.

**Country (Fallback)** You can set a fallback country in case a traffic sign does not exist for a newly chosen country.

## Global Road Attributes

**Track Width (m)** The width of the entire track is defined on both sides of the center line. On the entire width the friction coefficient is defined, e.g. 1.0 for dry asphalt or 0.2 for a slippery surface.

**Margin Width (m)** The margins are basically longitudinal stripes on both sides of the entire road length. For each left and right stripe you can assign a width and a friction coefficient. The width assigned will be added to the end of the track width on both sides of the road. The margins will be used only with a corner cutting coefficient > 1.0.

**Friction Stripe 1/2 (m)** In order to have a variable friction coefficient along the track width (applied on the entire track length), up to two stripes can be defined. As an example, this can be used for a  $\mu$ -split braking maneuver.



Figure 4.4: Example of two friction stripes with lower  $\mu$ .

Along those stripes, different friction coefficients to the main track can be applied.

The width and position of the stripes along the track width is parameterized in the field "from" and "to". The position is always considered besides the center line of the road: a positive value means on the left side, a negative value means on the right side.

If you want to insert a friction stripe that applies only for a certain segment on the road please refer to the [section 'Override Selected Segment Attributes' on page 47](#).

Having defined the global layout of the track, it has to be parameterized. As explained before, there are two ways to define the track layout:

- using digital measurements or
- building up a segment-based layout.

As examples take the TestRuns Examples > IntegratedControlSystems > ABS\_Braking\_musplit and ABS\_Braking\_FrictionPatchwork.



## GCS - Global Coordinate System

CarMaker is able to translate the position of the vehicle into WGS 84 coordinates, which can for example be used with Google Earth. As these maps are based on a global coordinate system whereas CarMaker uses cartesian coordinates, the vehicle position calculated by CarMaker needs to be transferred. For this, one point anywhere at the road needs to be given in both the cartesian coordinate system of the IPG virtual road frame (x, y, z) and in the corresponding geographic coordinate system on the earth surface as reference:

<b>GCS Lat (deg)</b>	Specifies the latitude of the geographic coordinates of the reference point.
<b>GCS Long (deg)</b>	Specifies the longitude of the geographic coordinates of the reference point.
<b>GCS Height (m)</b>	Specifies the elevation of the geographic coordinates of the reference point.
<b>GCS Projection Mode</b>	Two different methods which project the 3D ellipsoid graticule onto a 2D plane are available: Flat Earth or Gauss-Krüger. Please find more details on the differences in the appendix of the Reference Manual.
<b>Road Coord x,y,z (m)</b>	Specifies the global position of the reference point in the road frame Fr0.
<b>Coordinate Transformation</b>	
<b>Start Coordinates x,y,z (m)</b>	The start coordinates define the origin of the center line of the road.
<b>Start Direction (deg)</b>	In this field you can define the angle offset from the first segment regarding the x-axis.

#### 4.4.3 Segment Based Roads

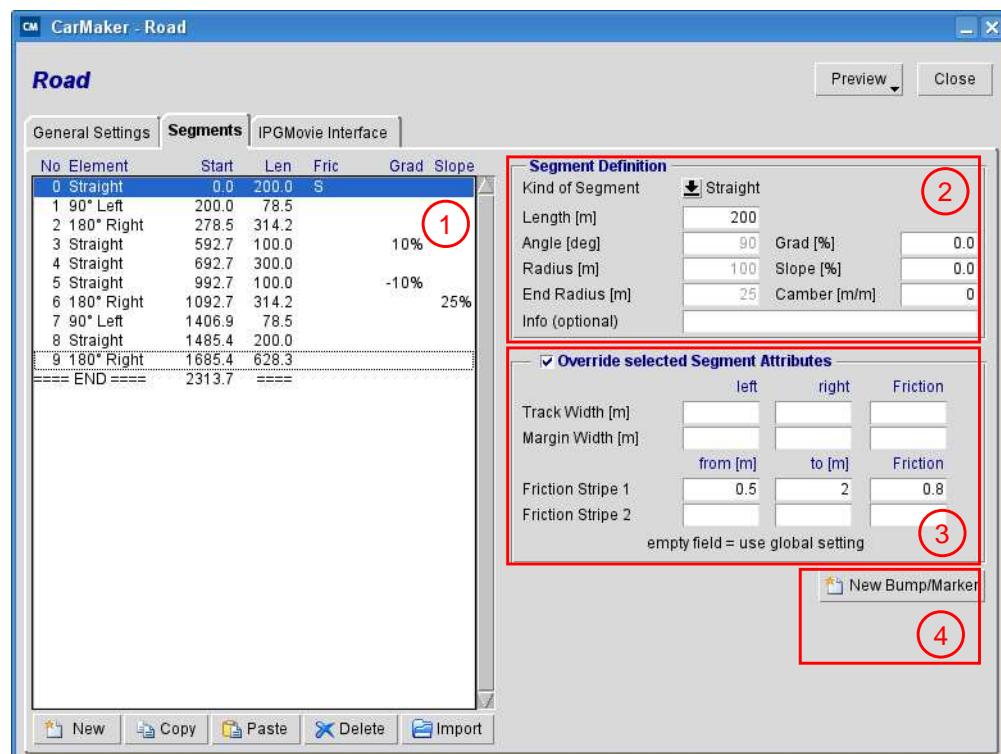


Figure 4.5: Segment-based road definition

The definition of the road geometry with a segment based approach consists of a succession of various segments (straight lines, curves left or right, or clothoids) with correctly parameterized properties in order to build a complete road. To use this method, move to the second tab of the IPGRoad dialog box called "Segments".

**Segment List:** The segment list is displayed and can be built in the box at the bottom left of the road GUI (see (1)). At this point, segments can be added or deleted. To add a segment to the end of the list, select the segment "==== END ====" (now it is highlighted in blue) and click on the button "New".

New  
Copy  
Insert  
Delete  
Import

If you wish to duplicate a segment, select the respective segment, click on "Copy", select the segment "==== END ====", and finally click on the button "Insert".

You also have the possibility to import the segments from another TestRun. For this, click on "Import", select the TestRun from which you want to import the road segments, and select "OK". Note that the imported segments are added to the list in this case; if other segments were already defined in the list, they are not deleted.

You can find the same functionality "Import" by clicking with the right mouse button anywhere in the Road-GUI. In this case you will also find an option "Delete all Segments". Otherwise, to delete only one segment, select it and click on the button "Delete".

### Segment Definition

Once you have built the list, you can define the exact 3D geometry of each segment. There-to see the box "Segment Definition", at the right of the Road GUI (2).

**Kind of Segment** It defines the general geometry of the segment selected in the segment list.

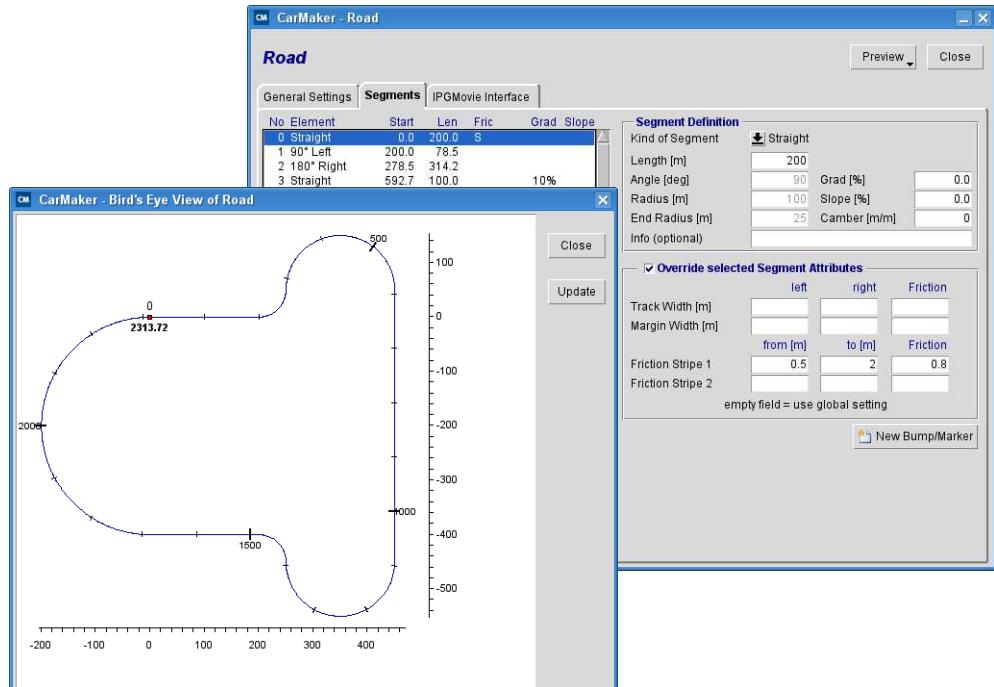
Table 4.1: Overview of segment kinds

Kind of Segment	Description
Straight	Straight line. Properties of this element: length, lateral and longitudinal slope, camber.
Turn left	Left curve. Properties of this element: angle, radius, lateral and longitudinal slope, camber.
Turn right	Right curve. Properties of this element: angle, radius, lateral and longitudinal slope, camber.
Clothoid left	Clothoid turning to the left. Properties of this element: angle, start and end radius, lateral and longitudinal slope, camber.
Clothoid right	Clothoid turning to the right. Properties of this element: angle, start and end radius, lateral and longitudinal slope, camber.

Note: a clothoid is a curve with varying radius. Its geometry looks like a snail shell.

**Length (m)** Parameter only used for a straight line.  
An informational value is given if another *Kind of Segment* is chosen.

- Angle (deg)** Parameters used to describe curves and clothoids. The curves use only the parameters *Angle* and *Radius*.
- Radius, End Radius (m)** For a clothoid, the value in *Radius* is used for the start radius, and the value *End Radius* for the end. Thus a smaller value in *End Radius* will lead to a curve that closes up like on the picture below:



**Grad, Slope (%)** Those parameters enable to define each selected segment in 3 dimensions.

- Camber (m/m)**
- *Grad* defines the longitudinal slope.
  - *Slope* defines the lateral slope.
  - *Camber* defines, in  $\frac{m_{Elevation}}{m_{LateralDeviation}}$  the shape of the road. It is used to create a parabolic transition of the road's lateral slope.

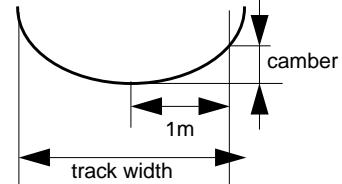


Figure 4.6: Definition of the camber

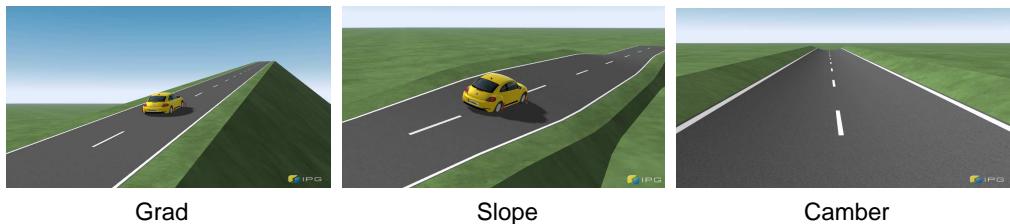


Figure 4.7: Examples for the effects of Grad, Slope and Camber



As an example take the TestRun Examples > CarMakerFunctions > IPGRoad > Segments\_Oval\_2D and Segments\_Oval\_3D.

- Info (optional)** A short description of the selected segment can be defined here.

### Override Selected Segment Attributes

Coming to box (3) *Override Selected Segment Attributes*. This option should be activated if you want to define a special width or friction coefficient for a single segment.

Those parameters are identical to the ones in the box *Global Settings*. Indeed they have exactly the same impact, at a few exceptions:

- Contrary to the *Global Settings*, the parameters under *Override* are all optional.
- If any value is defined, it is considered instead of the corresponding value at *Global Settings*.
- If you define a value, it will be applied to the selected segment only, not to the entire road length.

Subsequently, this option enables to define refined road properties. A quite common application would be a  $\mu$ -split braking TestRun, where you have a certain segment of the road which has a low friction stripe.

As an example take the TestRun Examples > VehicleDynamics > Rollover > FrictionStep.



### New Bump/Marker

The meaning of the button *New Bump/Marker* (4) will be explained in the oncoming [section 4.4.7 'Adding Road Markers' on page 58](#).

## 4.4.4 Digitized Roads

The second way to define a road in CarMaker is through loading digitized road data.

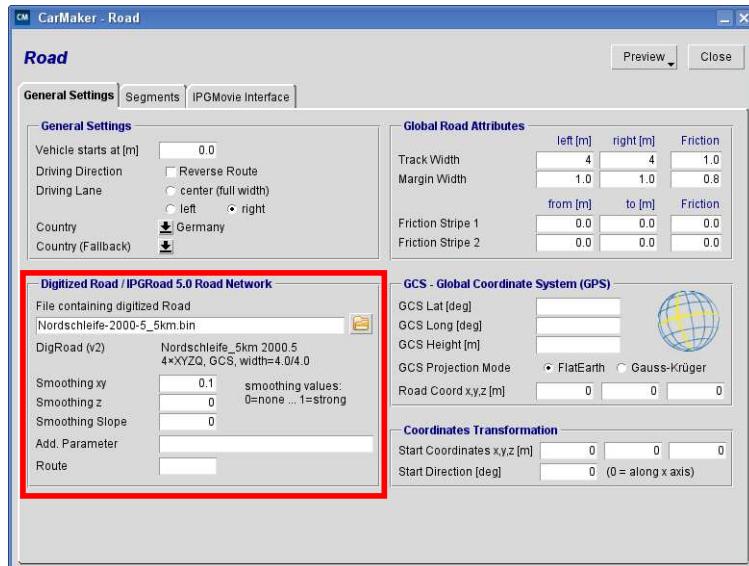


Figure 4.8: Loading digitized road data

#### File containing digitized Road

This parameter can be found in the tab *General Settings* of the IPGRoad dialog box. Here a digitized road profile can be loaded. Two different file types are available:

- KML files (WGS84-coordinates)

KML-files (Keyhole Markup Language), which are generated by Google Earth or Google Maps, can be imported directly. Please note, that the file extension must be \*.kml in this case. During the generation of the road, the WGS84-coordinates will be translated into cartesian coordinates. The smoothing of the road is carried out by an approximated spline.

To be readable for CarMaker, the KML-file needs to contain the following case-insensitive structure. For each point longitudinal and lateral geographic coordinates and elevation need to be provided.

**Example**

```
<placemark>
    <multigeometry> and / or <linestring>
        <coordinates>
            </coordinates>
        </multigeometry> and / or </linestring>
    </placemark>
```



If the kml-file contains the placemark-tag more than once, only the first one will be considered by Carmaker. But within the first tag, several multigeometries or linestrings can be available.

- ASCII files (cartesian coordinates)

The ASCII files contains tabulated data points. These points describe the course of the center line of the road in 3D either in the xy plane (x,y) or in GPS coordinates (long, lat). Including the altitude (z) and the slope of the road surface (q) and the track width for left and right side (wl/wr) is possible as well. Between two data points, a cubic interpolation is applied.



Both files only describe the general layout of the road. Other road characteristics like surface friction, left and right lane width, obstacles etc. are defined in the global road attributes or segment based if need be.

A digitized road file of type ASCII has a structure as displayed below:

**Example**

```
:      x      y      z      q      wl      wr
#
# IPG-ROADDATA DEMO
# Nuerburgring-Nordschleife DEMO (approx 5000m)
#
# Version          2000-1
# Status of the data   1999
#
# (c) 1999-2000
# IPG Automotive GmbH, Karlsruhe
#
# x      y      z      q      wl      wr
0.000  0.000  0.000  0.01111  4.10   3.90
0.480  0.000  0.009  0.01176  4.10   3.95
1.000  0.000  0.018  0.01236  4.05   4.00
1.540  0.000  0.028  0.01294  4.01   4.05
2.090  0.000  0.037  0.01350  4.00   4.10
2.630  0.000  0.046  0.01401  3.90   4.20
3.130  0.000  0.056  0.01452  3.70   4.20
...
```



You can replace the x and y tabulated data points by longitudinal and lateral geographic coordinates to define the road. To implement this, you must replace the first line starting with the column *x* and *y* with *long* and *lat*. Make sure to use at least eight decimal places in order to ensure a sufficient accuracy of the coordinates.

**Example**

```
: long lat z q
#
# IPG-ROADDATA DEMO
# Bremen-Oyten DEMO
#
# Version 2010-1
# Status of the data 2008
#
# (c) 1999-2000
# IPG Automotive GmbH, Karlsruhe
#
# long lat z q wl wr
8.804472000 53.09204560 0.000 0.01111 4.10 3.90
8.804329500 53.09217680 0.009 0.01176 4.10 3.95
8.804138700 53.09236090 0.018 0.01236 4.05 4.00
8.804016700 53.09249180 0.028 0.01294 4.01 4.05
8.803851800 53.09265810 0.037 0.01350 4.00 4.10
...
```



You can use the option *Reverse driving direction* to drive along the digitized road in the opposite direction (from the end to the beginning of the road). Please note, that the position of every bump/marker will be reversed, too.

### Definition Line

The first line indicated by a colon is the definition line. It rules the sequence in which the single components of each data point are stated. In the example above the first column contains the x coordinates of all data points, the second column contains the y coordinates, followed by the altitude z, the slope q.



The x and y coordinates (or instead long and lat) are compulsory, whereas the following columns can be left out. However, the sequence has always to be the same as in the example above. If you do not have any values for the altitude but you would like to define the lateral slope, you must not skip the column defining z. The column can consist of zeroes, but it needs to be defined.

### Comments

You can insert comment lines to the beginning starting with '#'.

In the example above some comments are added after the definition line to give some information about the road data.

Comment lines can be added at any point in the road data file.

### Data points

After the definition line and some comments, the data points of the digitized road are stated finally. Each data point has to be defined in a separate line. Within one line, the coordinates of the data points must be arranged in the same order as constituted in the definition line.

The data lines contain values separated by blanks or tabulators. Unit for the cartesian coordinates and the track width is [meter], slope is defined in [meter/meter], the geographic coordinates are given in degree (at least eight decimal places).

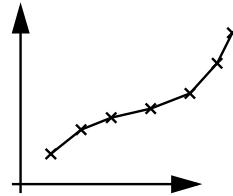


Figure 4.9: Entering the coordinates of the center line

The orientation is set in order to place the first two points on the x-axis of the coordinate system of the road. To prevent this, use the Additional Parameter "DisIsAbsolute" as explained below.

We recommend a point distance of 5-10m. Only in narrow corners, a higher resolution is required. IPGRoad uses a cubic spline interpolation between the digitized points.



In case you receive error messages referring to the resolution of your road or saying the track is too wide, you can use the smoothing parameters instead of manipulating your road data file. This is only necessary in case of major problems with the data points.

The smoothing parameters can be changed in the Road dialog. They only effect digitized roads following the old 4.5 description as explained above (not used for IPGRoad 5.0 road networks).

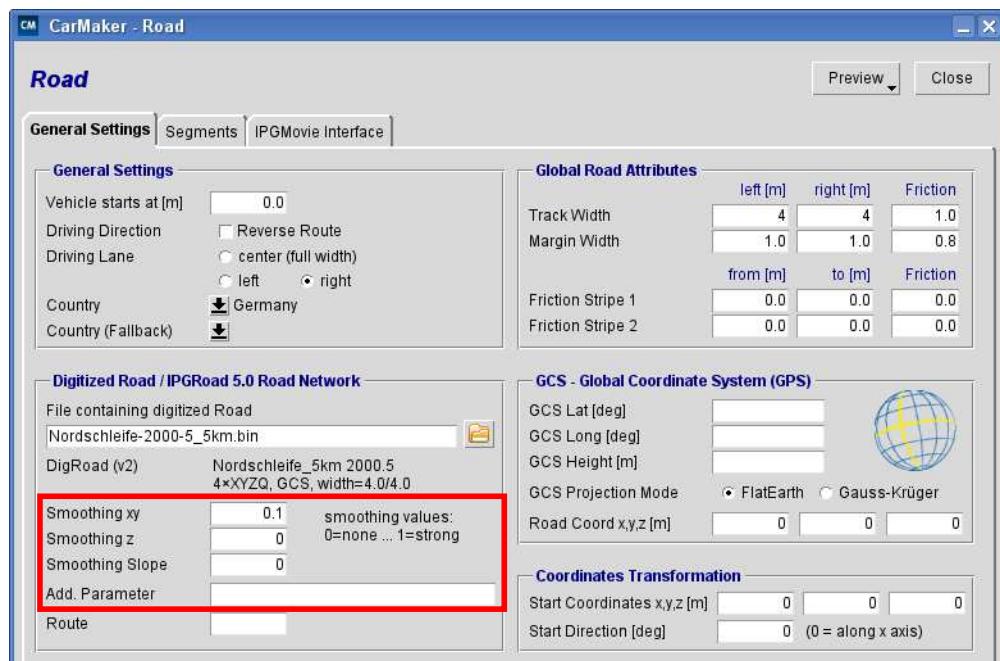


Figure 4.10: Smoothing parameters in the Road dialog

**Smoothing xy** The purpose of this parameter is to smooth measurement deviations in lateral direction. It smoothes the spline parameters of the road for xy-coordinates.

**Smoothing z** The purpose of this parameter is to smooth measurement deviations of the height profile. It smoothes the spline parameters of the road for the z-coordinate.

**Smoothing Slope** The purpose of this parameter is to smooth measurement deviations of the lateral inclination. It smoothes the spline parameters of the road for the q-coordinate.

**Add. Parameter** [Table 4.2:](#) includes the additional parameters for a digitized road

Table 4.2: List of Additional Parameters

Parameter	Description
MinDist	Filters points on the digitized road. Only points with the minimum gap defined here are used (no calculation of mean values).
useElev	Flag for the usage of elevation (z-) profile inside the digitized road. Default: 1
useSlope	Flag for the usage of the slope (q-) profile inside the digitized road. Default: 1
Verbose	Activate the verbose mode to get output in the Session Log for analyzation. To activate the mode set the parameter to 1.
digIsAbsolute	Key for digitized roads given in Cartesian coordinates. If added to the parameter list, the orientation of the digitized points will be absolute as given inside the file. The road cannot be translated or rotated anymore in the global frame. Default: 0 (expect for files which contain a GPS reference point like kml, here default is 1).

The parameter values are assigned by an equal sign.

**Example** MinDist=10

## Road Segments

All parameters in the box *Road Segments* (the list of segments and their properties), which have been described in the [section 'Segment Definition' on page 45](#) can be used for a digitized road, too.

In this case, the segments are used to define special properties of the road on a given distance. To be sure to reach the right distance, the *Kind of Segment* should always be a *Straight*, and its length orientates at the s - coordinate (road center line) of the digitized data.

If you wish, you can define longitudinal and lateral slope and camber. Of course, if you already have the z-coordinate in the file, it makes no sense to define another longitudinal slope than 0. If you do so, the longitudinal slope is added to the altitude defined in the file.

Finally, on a given distance you can also define a special width or friction coefficient with the option [section 'Override Selected Segment Attributes' on page 47](#).



The TestRun Examples > CarMakerFunctions > IPGRoad > Nordschleife\_RoadFeatures illustrates the use of the segments together with a digitized road. You can load this TestRun and look at the definition of each segment. At this point, we recommend to look especially at the first four segments. When you know which properties these segments have, start the TestRun to observe their effects.

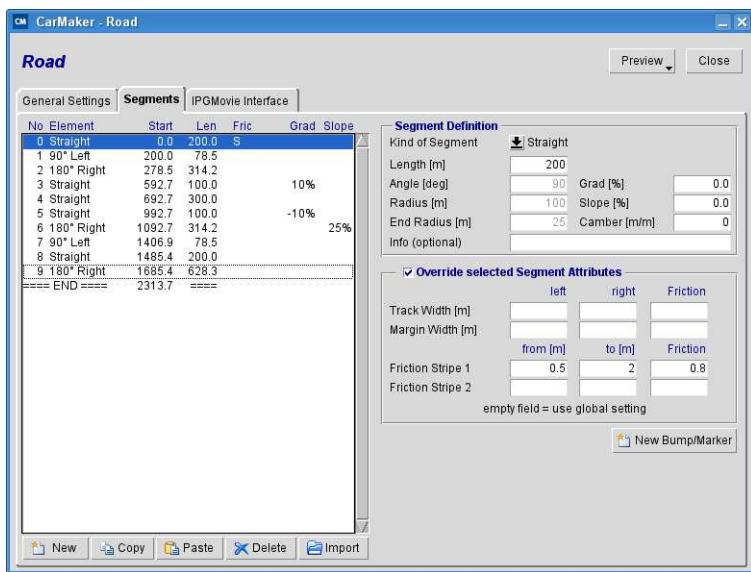


Figure 4.11: Overwriting segment attributes of a digitized road

#### 4.4.5 IPGRoad 5.0: Road Networks

You can find several example TestRuns with complex road networks under *Examples/IPGRoad5*. It is not recommended to edit these files except for the route definitions. The network can contain several different routes that are used by the test car or the traffic objects.



Road 5.0 networks can be exported using the third party tool ADAS RP by HERE. For a complete description, please refer to [section 10.5 'Road: CarMaker - ADAS RP Interface'](#)

To select a route for the test car, choose the route ID in the Road dialog under General Settings. See box 1 in the following picture:

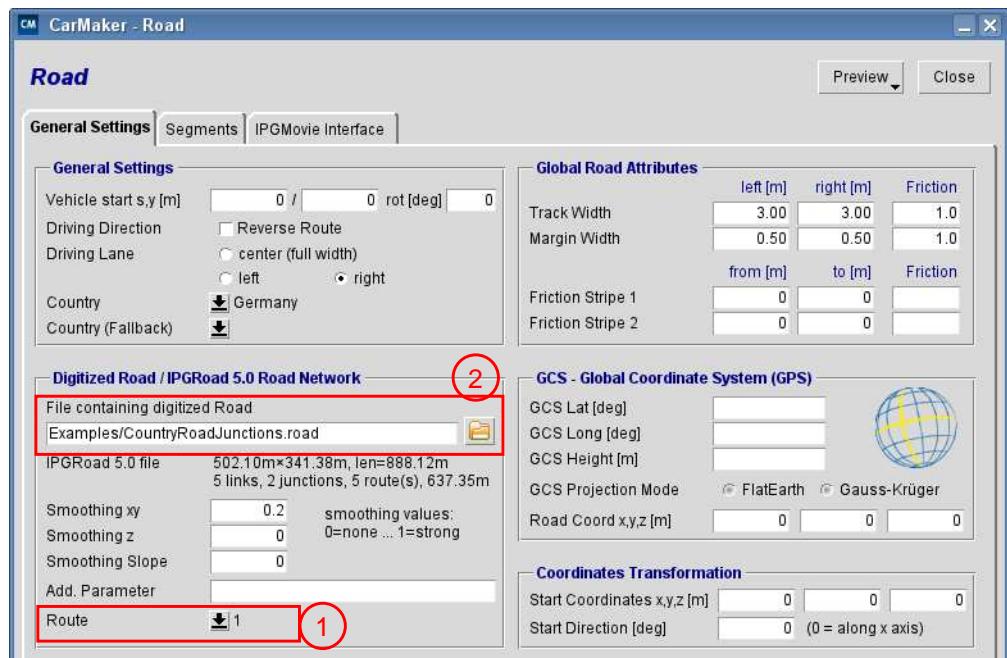


Figure 4.12: Route selection for the test car

Please find further information about the route selection for traffic objects in [section 'Start Conditions'](#).

The possible routes can be defined in the road configuration file which contains the road network (see box 2 in [Figure 4.12](#)). The routes are listed at the end of the file starting with the prefix "Route.<ID>.\*", as the following example shows:

```
Route.0:
    4
    1
    3
Route.0.Name = Route_00
Route.0.Path.0:
    0.0000 0 0
    45.0000 0 0.5
    85.0000 0 -0.3
    95.0000 0 0
Route.0.Path.1:
    0.0000 -3 0
    350.0000 -3 0
    372.0000 -3 0
    392.0000 -2 0
    408.0000 -2 0
Route.0.Path.2:
    0.0000 -2 0
```

**Route.<i>** The first key defines the order of links which are followed in a row. The route ID has to start with zero and counts upwards from there. The list of link IDs has to begin with a tab at the start of the line according to the common Infofile syntax.

**Route.<i>.Name** Define a name string for the route. It will be displayed in the Bird's Eye View (see [section 4.4.6 'Checking Road Geometry'](#)) to select the route of interest.

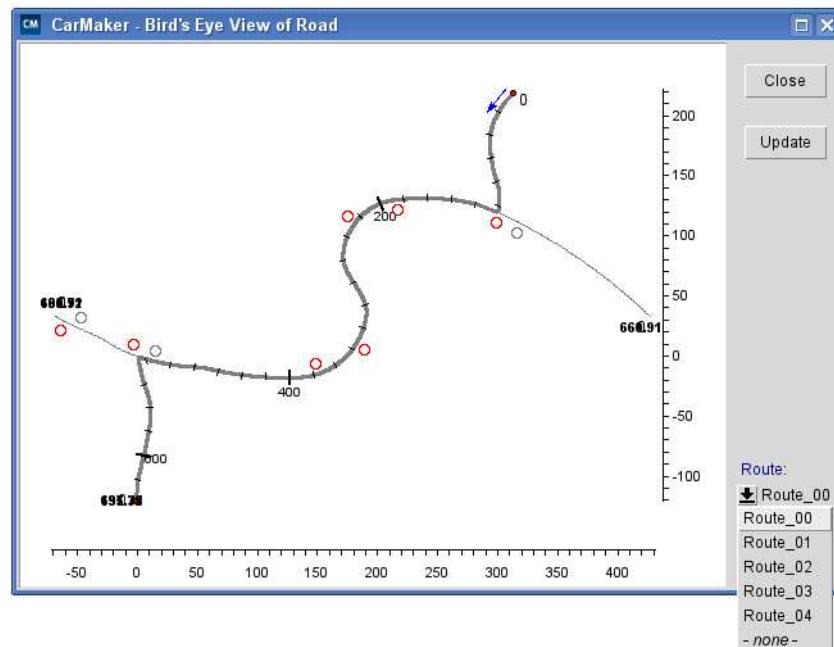


Figure 4.13: Display of different routes in the Bird's Eye View

**Route.<i>** This key is optional. In case the route path is not specified, the driving path for IPGDriver and traffic objects orients at the global lane settings defined in the Road dialog (see section 'Driving Lane'). Otherwise, a path along a route can be specified individually for each link (j). The first column in this key defines the s-coordinate (along route) which is followed by the lane number and the lateral offset from the lane center in meter. The lane numbering is defined as follows in IPGRoad:

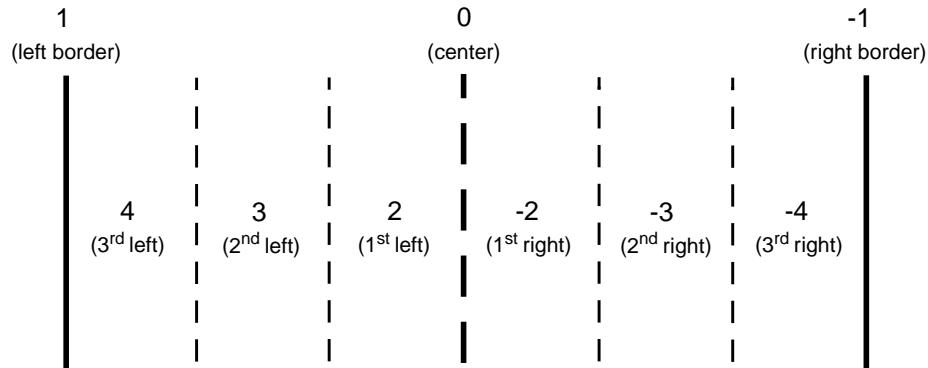


Figure 4.14: Numbering of lanes in IPGRoad

#### Example

A lane change maneuver from the right to the middle lane could look as follows:

Route.0.Path.0:

```
0.0000 -4 0
350.0000 -4 0
372.0000 -3 0
400.0000 -3 0
```

## 4.4.6 Checking Road Geometry

The Road dialog offers several preview modes, to check the road geometry and its surroundings. To choose a mode, keep the left mouse button pressed on the *Preview* button in the top right corner of the dialog.

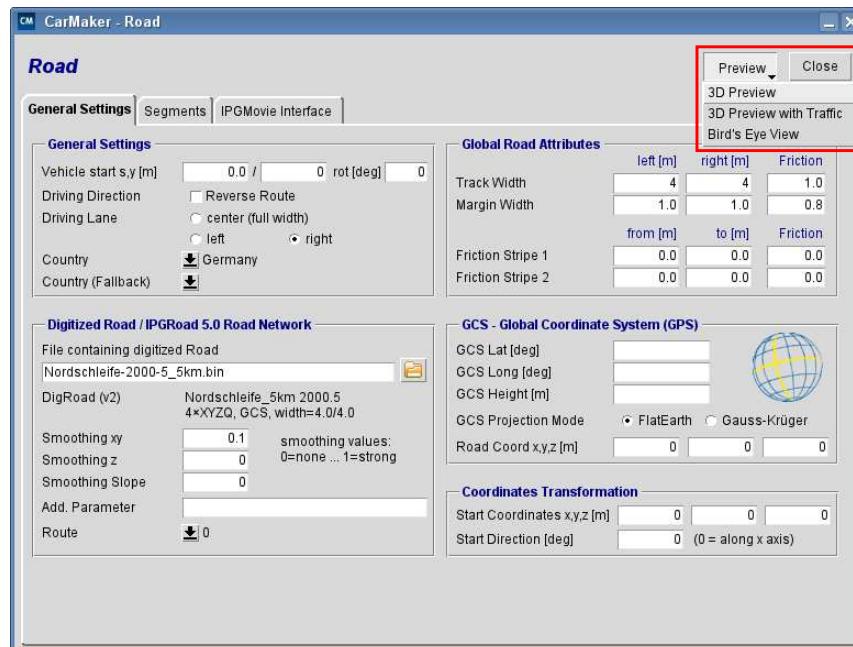


Figure 4.15: Activating the road preview

## Birds Eye View

The *Bird's Eye View* is a simplified preview which shows the road from top.

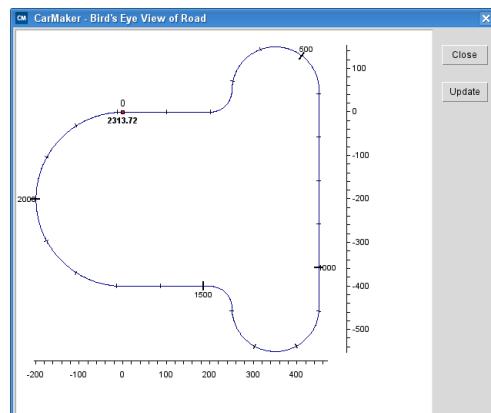


Figure 4.16: *Bird's Eye View* mode of the Road Preview

If you make any change to the road geometry, you just have to click on *Update* in the Bird's Eye View window in order to see the change. The segment which is currently selected in the list of road segments is highlighted in blue. The same goes vice versa: You can select another segment in the Bird's Eye View and the corresponding segment in the list of road segments will be active.

## 3D Preview

Whereas the Bird's Eye View only shows you the general course of the road in a flat perspective, the 3D Preview gives you the 3D representation of the track you built.

Once you selected the 3D Preview, IPGMovie will pop up. The camera view is moving along the road showing a yellow ball that follows the center line.

The 3D Preview includes the layout of the track regarding course and width as well as road features like bumps, markers or elements of the user defined environment.

The preview mode of IPGMovie is indicated by the words *Road Preview* at the bottom left corner of IPGMovie and can be started by clicking on the play button in the footer menu. Please find further information on the handling of IPGMovie in [section 7.2.2 'Replay' on page 403](#).

To visualize any changes to the road, simply click once again on the *Preview* button in the Road dialog.

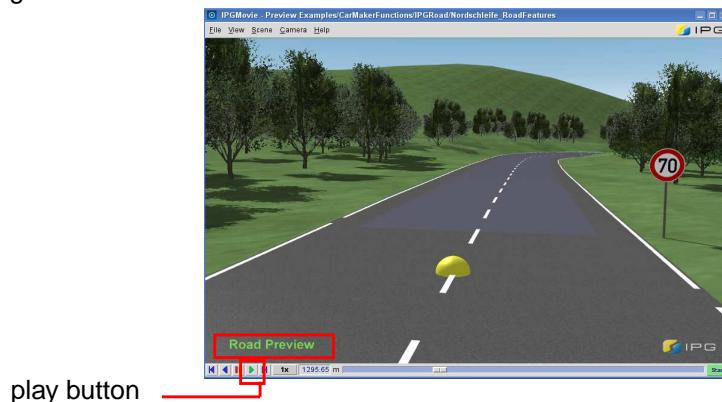


Figure 4.17: 3D Road Preview in IPGMovie

## 3D Preview with Traffic

This is the same as 3D Preview, but in this mode all traffic objects will be displayed additionally. Please find more information about the positioning of traffic objects in [section 4.8 'Traffic' on page 127](#).

## Special Preview modes

By clicking right in the IPGMovie windows, several display options are available for the 3D preview mode.

- Normal** The option *normal* shows the road and environment in the same style as later in the animation.
- Technical Drawing** This is a special design model. It shows the elements of the road with numbers, to indicate the driving path, sections and links with different colors.

Table 4.3: Color codes of technical Road Preview

Color	Description
red	Indicates the $n^{\text{th}}$ node of the path link, naught argument at beginning of road
black	Indicates the $n^{\text{th}}$ node of the x,y-coordinates (data points in case of a digitized road), naught argument at beginning of road
dark blue	Indicates the $n^{\text{th}}$ node of the height profile, naught argument at beginning of road
light blue	Indicates the $n^{\text{th}}$ node of the camber profile, naught argument at beginning of road
pink	Indicates the $n^{\text{th}}$ node of the slope profile, naught argument at beginning of road

- Wireframe** This feature is available for both options, the normal and the technical preview. It shows the elements used to construct the road.

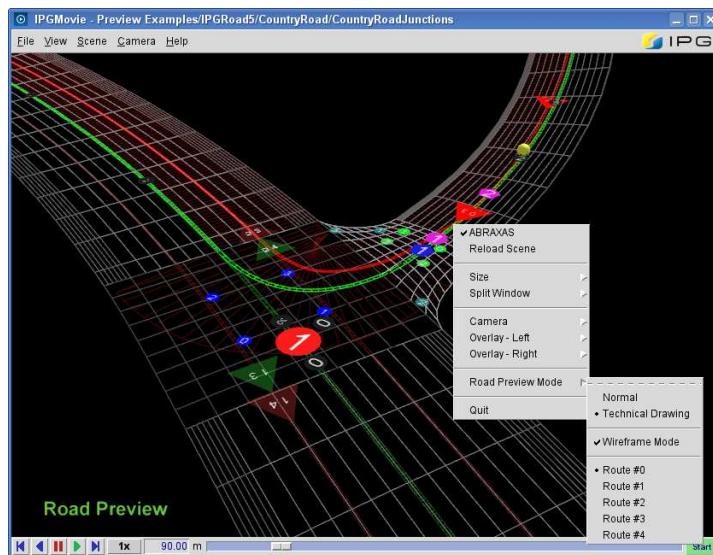


Figure 4.18: Changing the display mode of the Road Preview via right-click

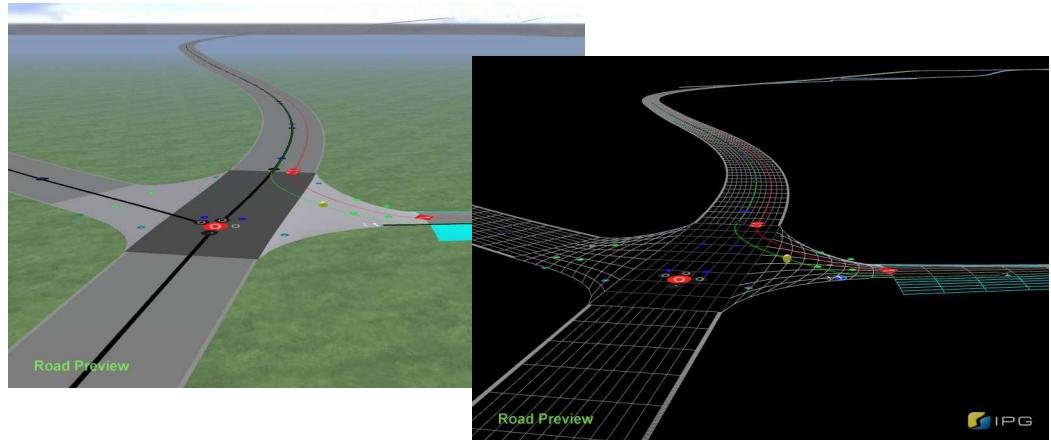


Figure 4.19: Technical Road Preview with (right) and without (left) wireframe

## 4.4.7 Adding Road Markers

On the road, bumps and markers can be inserted to modify the track's appearance. All these options are available in the *Segments* tab of the Road dialog under *Bumps/Markers/Movie Objects*. CarMaker differentiates four groups of road markers depending on their effect on the simulation:

- *Markers* influence the behavior of IPGDriver, the driving maneuver or the vehicle. Some of them can be detected with the Road Property Sensor (see [section 5.23.7 'Road Property Sensor'](#)).
- *Bumps* describe a local modification of the road surface.
- *Sensor Objects* can be detected with one of CarMaker's sensor models (see [section 5.23 'Sensors'](#)).
- *Movie Geometries* may be used for beautification of the environment displayed in IPG-Movie. They have no influence on the simulation (e.g. they are invisible for the sensors).

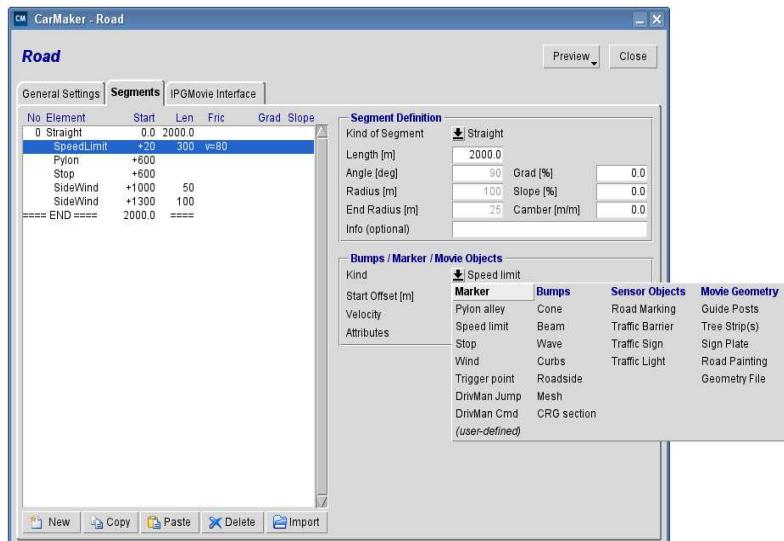


Figure 4.20: Selecting segment based bumps or markers



All *Markers*, *Bumps*, *Sensor Objects* and *Movie Geometries* (except Geometry files) should always be placed on the road or the road margin. Choosing a position that is outside the margin can lead to unexpected positioning.

### Parameterizing the Road Markers

On the road, markers can be inserted which add some indication to the rest of the environment. All these markers are available for the global track as well as for a road segment only.

Markers are valid for a certain section of the road. This stretch defined by the marker obtains additional attributes depending on the marker type.

CarMaker offers the following markers:

- *Pylon Alley* for the simulation of pylon courses.
- *Speed Limit* for the simulation of velocity signs.
- *Stop* to let the car stop, e.g. at a crossover.
- *Wind* for the simulation of wind impacts.
- *Trigger Point*, for reading a given UAQ at a given distance on the road.

- *DrivMan Cmd*, to insert a minimaneuever command.
- *User defined*, to insert a marker with user defined parameters.

The beginning of a Marker is specified by the arc length. The end of the marker section is defined by a length counting from the start point.

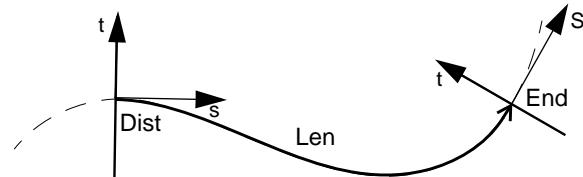


Figure 4.21: Range of a Marker

Additional marker specific parameters may be specified.

**Pylon Alley** Using the *Pylon Alley* marker, pylons can be defined:

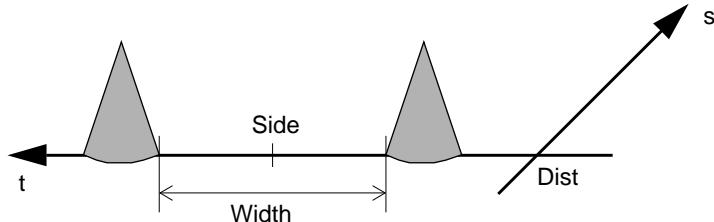


Figure 4.22: Definition of Pylon Alley markers

A *Pylon Alley* marker is defined in the *Bumps/Markers* dialog by the following parameters:

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
y Offset [m]	Lateral offset from the centerline (measured orthogonally to the direction vector).
Width [m]	Width between the two pylons.
Attributes	Invisible: With this option activated, the pylons are not displayed in IPGMovie but still considered by IPGDriver.

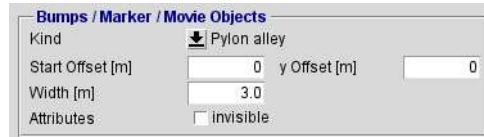


Figure 4.23: Pylon Alley definition in the Road dialog



Figure 4.24: Using a Pylon Alley for a slalom maneuver



As an example take the TestRun Examples > VehicleDynamics > Slalom\_18m and Slalom\_36m.

#### **Speed Limit**

Using this marker, a speed limit sign can be set which is valid for a certain length or up to the next sign. Please note that the *Traffic Sign Sensor* does not detect this kind of speed limit.

A *Speed Limit* marker is defined in the *Bumps/Markers* dialog by the following parameters:

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
Length [m]	Optional. Length of the speed limit section. A limit clearing sign automatically is inserted after the specified section length. If no length is defined, the speed limit is valid up to the next Speed Limit marker.
Velocity	Maximum velocity allowed (integer, decimal will be cut).
km/h / m/s	Unit of the velocity.
Attributes	Invisible: With this option activated, the speed limit is not displayed in IPGMovie but still considered by IPGDriver.

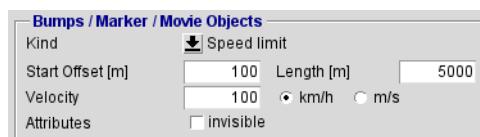


Figure 4.25: Specifying a speed limit in the Road dialog



Figure 4.26: Speed limit and limit clearing



As an example take the TestRun Examples > CarMakerFunctions > IPGRoad > Road\_Markers.

- Stop** Using this marker, the vehicle can be brought to standstill. It will be stopped at the sign and stay for a user defined time span. Please note that the *Traffic Sign Sensor* does not detect this kind of stop sign.

A *Stop* marker is defined in the *Bumps/Markers* dialog by the following parameters:

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
Duration [s]	The duration the car is standing still.
Attributes	Invisible: With this option activated, the stop sign is not displayed in IPGMovie but still considered by IPGDriver.

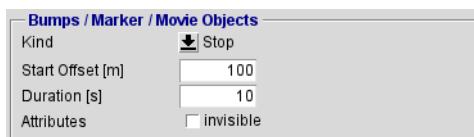


Figure 4.27: Defining a Stop marker in the Road dialog

The *Stop* marker is indicated in IPGMovie using a stop sign and a textual information about the time which is left until the car starts driving again.



Figure 4.28: Visualization of the remaining stop time in IPGMovie



As an example take the TestRun Examples > CarMakerFunctions > IPGRoad > Road\_Markers.

#### Wind

This marker generates wind on a given distance of the road. The wind has some properties that you can parameterize, like its speed and direction. When the vehicle passes the wind or, more accurately, when the point *Aero Marker* of the vehicle (see the section "Aerodynamics > General Parameters" in the Reference Manual) passes the wind, the additional forces corresponding to the interaction of the wind and the vehicle aerodynamics are calculated.

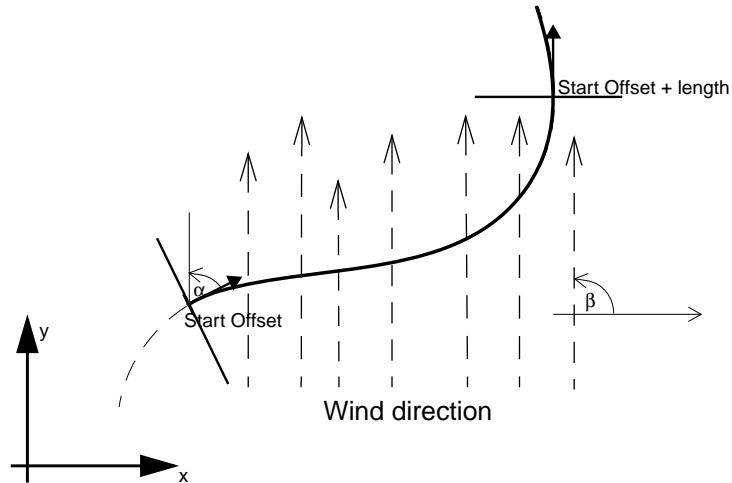


Figure 4.29: Definition of Wind Markers

A *Wind* marker is defined in the *Bumps/Markers* dialog by the following parameters:

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
Length [m]	Optional. Length of the wind stretch. Otherwise wind is blowing up to the next Wind marker.
Velocity	Velocity of the wind.

Setting	Description
Direction [deg]	Wind direction $\alpha$ . A value of zero equals back wind.
Attributes	Invisible: With this option activated, the side wind area is not indicated by flags in IPGMovie but still considered by IPGDriver.

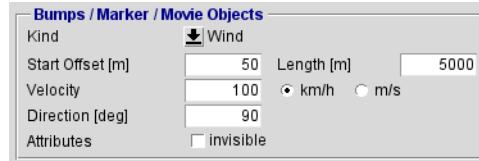


Figure 4.30: Definition of a wind attack in the Road dialog



Figure 4.31: The wind is indicated by flags at the road margin

**Trigger Point** Using the road marker *TriggerPoint*, different actions can be executed at the given road position *sRoad*.

Currently time and distance measurements are implemented. This means, that at a given distance of the road (variable *sRoad*), you can read the current time. You can observe this value in IPGControl under *DM.TriggerPoint.Time*.

Furthermore, the distance travelled between two trigger points is recorded by the quantity *DM.TriggerPoint.Dist*.

The syntax to define a *TriggerPoint* in the *Bumps / Markers* menu is the following:

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
Mode	Mode = 1: define new reference point Mode = 2: take all measurements Mode = 3: take all measurements and define new reference point.
Id	This is a positive integer value and should be numbered consecutively.

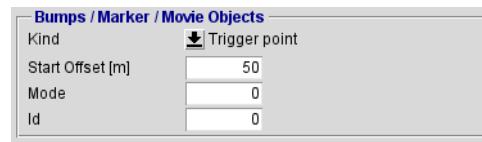


Figure 4.32: Creating a trigger point in the Road dialog

The measurements at the trigger points can be plotted in IPGControl or section and laptime measurements can be integrated in the Instruments window. A very useful application of *TriggerPoints* is to take the sector times on racing tracks.

- DrivMan Jump** The option *DrivMan Jump* in the *Bumps / Markers* dialog gives the opportunity to enter a specified maneuver with a start offset of the current road segment as the trigger point.

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
Target Label	Label of the maneuver target.

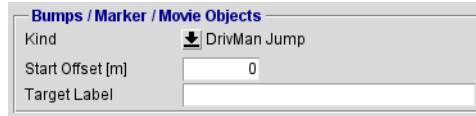


Figure 4.33: Dialog to enter road based minimaneuver commands

- DrivMan Cmd** The option *DrivMan Cmd* in the *Bumps / Markers* dialog gives the opportunity to enter a minimaneuver command with the road distance as trigger. All commands of the Minimaneuver Command Language can be used. Please find a description of all commands in [section 'Minimaneuver Command Language' on page 590](#). Required parameters are:

Abbreviation	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
Minimaneuver Command	Minimaneuver command string.

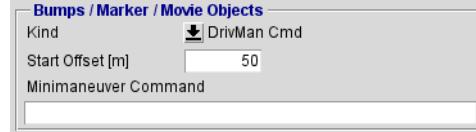


Figure 4.34: Dialog to enter road based minimaneuver commands

**Example** You can create a log message to the Session Log after a certain distance of the road was travelled by the vehicle:  
 Start Offset: 100m  
 Minimaneuver Command: LogMsg "The car has travelled 100m."

**Example** You can trigger a braking maneuver after a certain point on the road:  
 Start Offset: 300m  
 Minimaneuver Command: DVArw DM.Brake Abs DM 3000 0.7

### User Defined



User defined markers can be used to place a marker with a list of user defined parameters along the road. The parameters can be of any type: integer, float, double or string.

The user defined marker can be detected by a *Road Property Sensor* (see [section 5.23.7 'Road Property Sensor' on page 292](#)). For each parameter, a new quantity called *RPSensor.Name.RMarker.Attr.\** is created. Please note, that the Road Property Sensor can detect a maximum amount of 10 attributes which are numbers (strings are skipped in the parameter list).

Required parameters for the definition of a User Marker are:

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
Length [m]	Extent of validity for this marker.
Marker Name	Name of the marker, as defined in the code.
Parameters	Several parameters can be assigned, the character format needs to be specified in the marker declaration in the c-code.

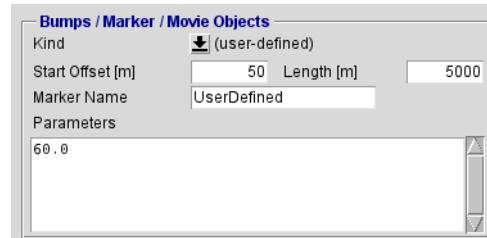


Figure 4.35: Parameterization of a user defined marker

## 4.4.8 Adding Road Bumps

Bumps can give a three-dimensional layout to the road surface. They can be applied to a segment based road design as well as to a digitized road profile. To insert a new bump proceed as explained in the beginning of the previous [section 4.4.7 'Adding Road Markers' on page 58](#).

CarMaker supports the following types of bumps:

- Cone, to add frustums with an elliptical base to your road.
- Beam, to insert a barrier with a rectangular base.
- Wave, to put a waved obstacle on your road.
- Curbs, to insert curbs besides the road.
- Roadside, to insert sidewalks or ditches with varying heights besides the road.

- Mesh, to insert a freely definable obstacle with a rectangular base.
- CRG Section, to import measured road data into your simulation.

**Cone** Cones are frustums with an elliptical base.

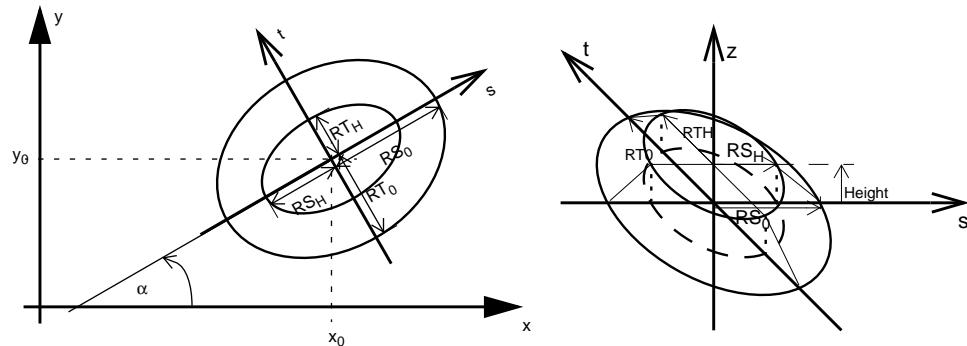


Figure 4.36: Definition of Cone Obstacles

A Cone is defined in the *Bumps/Markers* dialog by the following parameters:

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
y Offset [m]	Lateral offset from the centerline (measured orthogonally to the direction vector).
Rotation [deg]	Angle of the cone relatively to the road direction vector s.
Height [m]	Total height of the cone.
Radius / Radius 2 [m] long/lat	Radii of the cone. The smaller one automatically defines the upper side of the cone.
Friction	Friction coefficient of the whole obstacle.
Material/Texture	Insertion of: <ul style="list-style-type: none"> <li>RGB-colors and an optional alpha-channel with a range from 0-1.0 (e.g. 0.2, 0.6, 1.0, [0.5])</li> <li>Textures from a file</li> </ul>

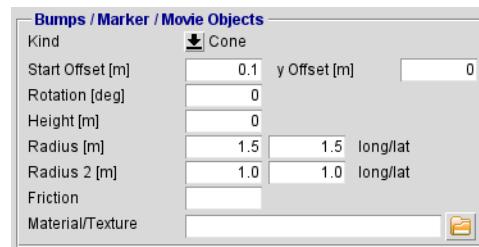


Figure 4.37: Definition of a cone in the Road dialog



Figure 4.38: Effect of a cone in IPGMovie

**Beam** A *Beam* is a barrier with a rectangular base. The overall length results from the sum of three partial lengths. The width of the obstacle has to be specified. The beam automatically adjusts itself to the course of the road.

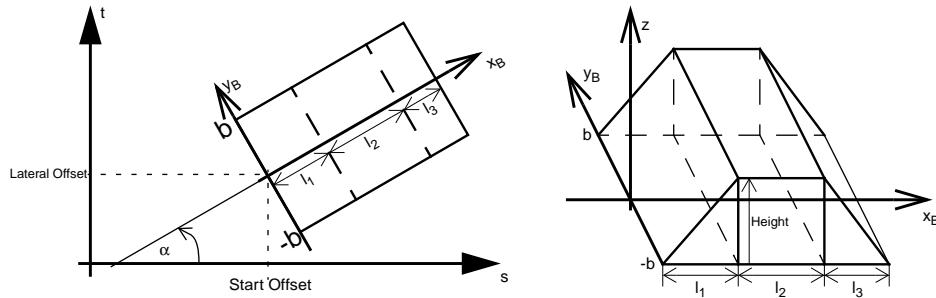


Figure 4.39: Definition of Beam Obstacles

A *Beam* is defined in the *Bumps/Markers* dialog by the following parameters:

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
y Offset [m]	Lateral offset from the centerline (measured orthogonally to the direction vector).
Rotation [deg]	Angle of the beam relative to the road direction vector s.
Height [m]	Total height of the beam.
Width [m]	Width of the beam.
Length up [m]	Length of the upward ramp.
Length top [m]	Length of the plateau.
Length down [m]	Length of the downward ramp.
Friction	Friction coefficient of the whole obstacle.
Color/Texture	Insertion of: <ul style="list-style-type: none"> <li>RGB-colors and an optional alpha-channel with a range from 0-1.0 (e.g. 0.2, 0.6, 1.0, 0.5)</li> <li>Textures from a file</li> </ul>

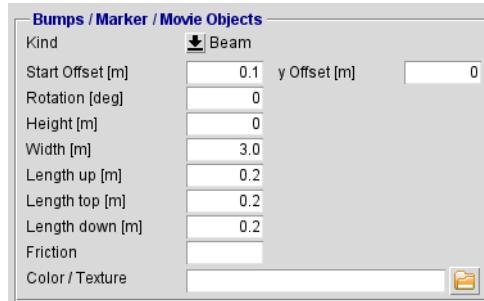


Figure 4.40: Definition of a beam in the Road dialog



Figure 4.41: Effect of a large beam in IPGMovie

**Wave** The obstacle *Wave* is defined by a rectangular base with *Length* along the road s-axis and *Width* along the t-axis. The wave automatically adjusts itself to the course of the road.

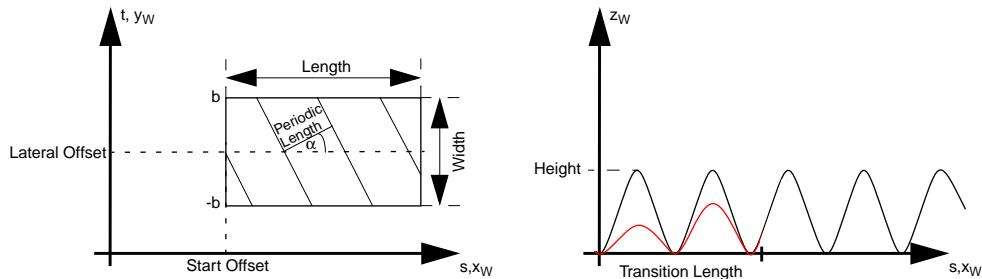


Figure 4.42: Definition of Wave Obstacles

A Wave is defined in the *Bumps/Markers* dialog by the following parameters:

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
y Offset [m]	Lateral offset from the centerline (measured orthogonally to the direction vector).
Length [m]	Total length of the wave.
Rotation [deg]	Angle of the wave(s) relatively to the direction vector.
Height [m]	Total height of the wave.
Width [m]	Width of the wave.
Period length [m]	Length of one period.

Setting	Description
Transition length [m] long/lat	The longitudinal transition length is the length within which the height is reached from the starting edge of the wave in the longitudinal direction. The lateral transition length is the length within which the height is reached from the lateral edge of the wave.
Friction	Friction coefficient of the whole obstacle.

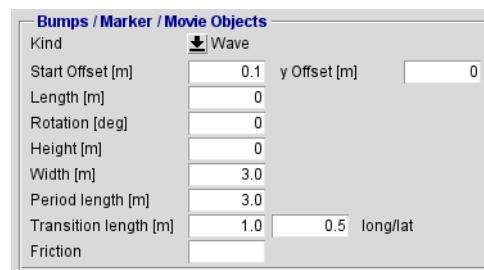
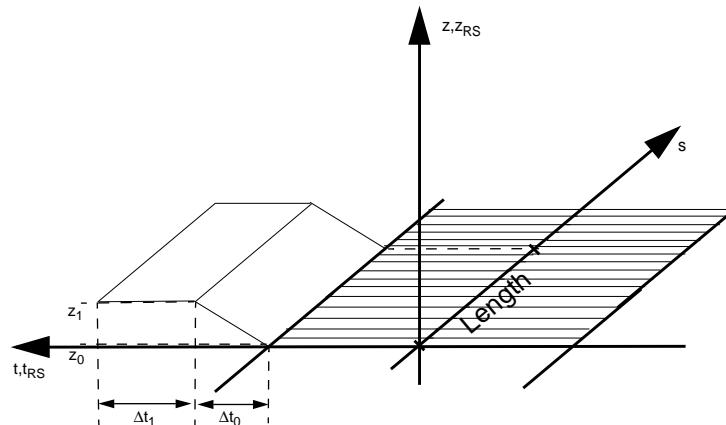


Figure 4.43: Definition of a wave in the Road dialog



Figure 4.44: Effect of waves in IPGMovie

**Curb** A *Curb* bump can be used to display simple objects at the side of the road which can be defined with one height and one width like a curb on a race track.



A *Curb* is defined in the *Bumps/Markers* dialog by the following parameters:

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
Length [m]	Total length of the curb.
Road side	Choose on which side of the road the curb should be placed; on the right, the left or on both sides.
Height [m]	Total height of the curb.
Width [m]	Width of the curb.
Width top [m]	Width of the top of the curb.
Friction	Friction coefficient of the whole obstacle.
Color/Texture	Insertion of: <ul style="list-style-type: none"> <li>RGB-colors and an optional alpha-channel with a range from 0-1.0 (e.g. 0.2, 0.6, 1.0, 0.5)</li> <li>Textures from a file</li> </ul>
Texture length	Stretching/compressing of the texture

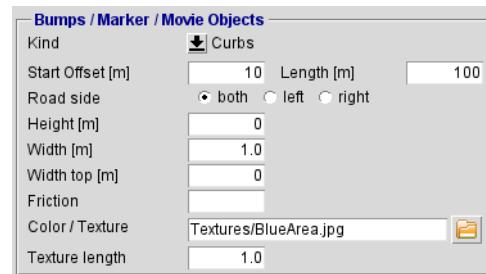
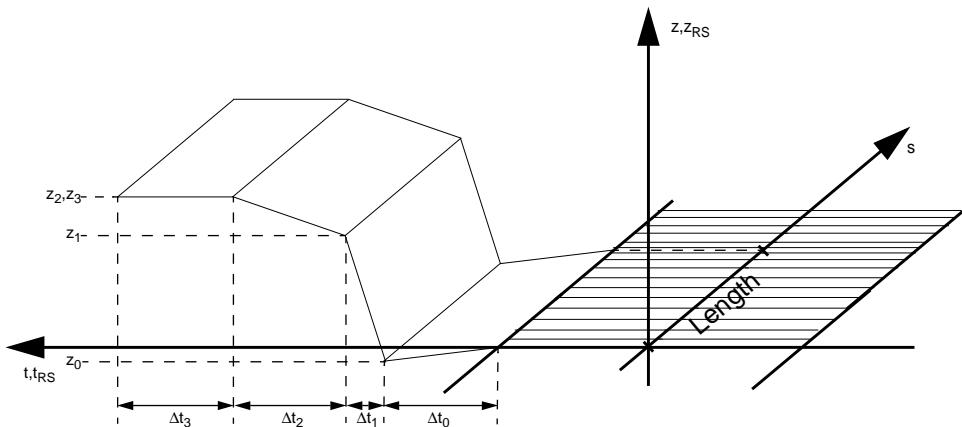


Figure 4.45: Definition of curbs in the Road dialog



Figure 4.46: Visualization of curbs in IPGMovie

**Roadside** A *Roadside* bump can be used to display complex objects at the side of the road which needs to be defined with different heights and widths like a sidewalk.



A *Roadside* is defined in the *Bumps/Markers* dialog by the following parameters:

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
Length [m]	Total length of the roadside.
Road side	Choose on which side of the road the roadside should be placed; on the right, on the left or on both sides.
Friction	Friction coefficient of the whole obstacle.
Texture length	Stretching/compressing of the texture along s-axis
Color/Texture	Insertion of: <ul style="list-style-type: none"> <li>RGB-colors and an optional alpha-channel with a range from 0-1.0 (e.g. 0.2, 0.6, 1.0, 0.5)</li> <li>Textures from a file</li> </ul>
Point list	Settable points for the roadside in $\Delta t$ and $z$ coordinates ( $\Delta t >= 0$ ) $\Delta t_0 \ z_0$ $\Delta t_1 \ z_1$ $\Delta t_2 \ z_2$ $\dots$ $\Delta t_n \ z_n$

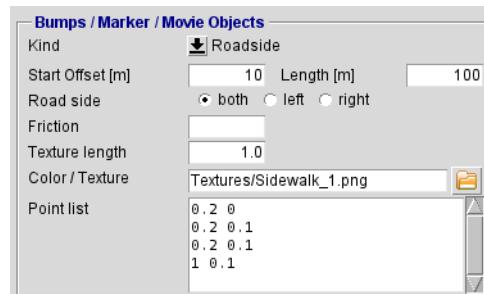
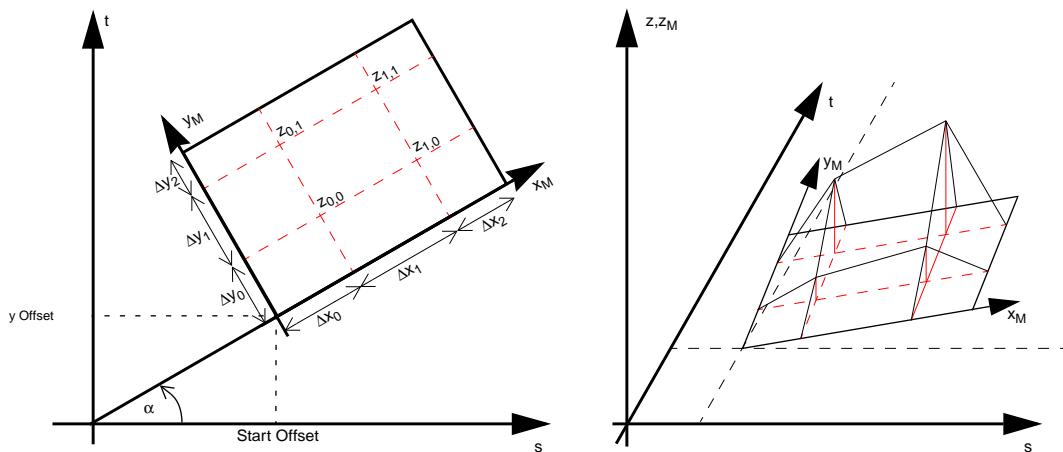


Figure 4.47: Definition of a roadside in the Road dialog



Figure 4.48: Display of a roadside in IPGMovie

**Mesh** A *Mesh* is an obstacle with a rectangular base. In general, the mesh is parameterized by a point list and some optional additional parameters. The overall length results from the sum of the partial lengths.



It is recommended to check the mesh parametrization using the "Wireframe Mode" inside the Road Preview of IPGMovie. See [section 4.4.6 'Checking Road Geometry'](#) in this chapter.

A *Mesh* is defined in the *Bumps/Markers* dialog by the following parameters:

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline). The start offset is measured to the rear, right point of the mesh.
y Offset [m]	Lateral offset from the centerline (measured orthogonally to the direction vector).
Rotation [deg]	Angle of the mesh relative to the direction vector of the road.
Mode	If <i>rectangular</i> is selected, the mesh will stay in its rectangular shape even in curves.
Friction	Friction coefficient of the whole mesh.

Setting	Description
Color/Texture	<p>Insertion of:</p> <ul style="list-style-type: none"> <li>RGB-colors and an optional alpha-channel with a range from 0-1.0 (e.g. 0.2, 0.6, 1.0, 0.5)</li> <li>Textures from a file</li> </ul>
Point list	<p>Defines the dimension of the mesh in meters. The values in the first row define the width of the lateral sections from the right to the left. The last value in the first row defines the distance from the outer left side to the ground.</p> <p>The first value within the next rows defines the length of the respective section. The following values define the height of the points of intersection with the width of the lateral sections.</p> <p>The single value in the last row defines the length of the mesh from the last point of intersection back to the ground.</p> <p> <math>\Delta y_0, \Delta y_1, \Delta y_2, \dots, \Delta y_n</math>  <math>\Delta x_0, z_{0,0}, z_{0,1}, \dots, z_{0,n-1}</math>  <math>\Delta x_1, z_{1,0}, z_{1,1}, \dots, z_{1,n-1}</math>  ... <math>\Delta x_{m-1}, z_{m-1,0}, z_{m-1,1}, \dots, z_{m-1,n-1}</math>  <math>\Delta x_m</math> </p>
Additional Parameters	<p>The following additional parameters are added at the beginning of the point list text box.</p> <ul style="list-style-type: none"> <li><i>tType</i>: defines the interpolation method between the points 1: bilinear (default), 2: steps (sample &amp; hold), 0: bicubic</li> <li><i>uScale</i> = 1 (default) scales the mesh in s-road direction</li> <li><i>vScale</i> = 1 (default) scales the mesh in y-direction</li> <li><i>zScale</i> = 1 (default) scales the elevation profile</li> </ul>

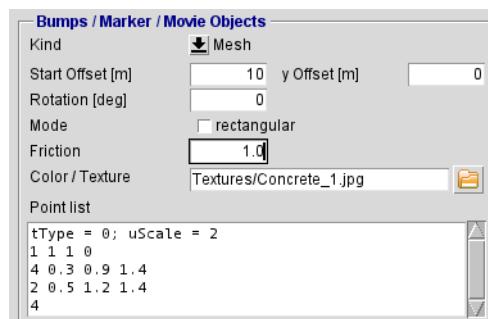


Figure 4.49: Definition of a mesh in the Road dialog

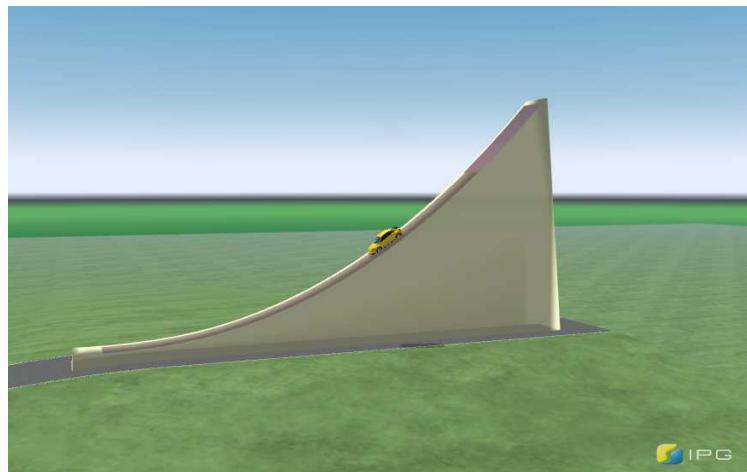


Figure 4.50: Effect of a mesh in IPGMovie



As an example take the TestRun Examples > VehicleDynamics > Rollover > Ramp.

#### CRG Section

The CRG section allows the user to import three-dimensional surface measurements in the \*.crg format developed by DaimlerChrysler Research and Technology and TÜV Süd. The CRG (curved regular grid) road data is gained from a special equipped vehicle for high-resolution measurements of road surfaces in the moving traffic.

To use this data you need to save the CRG data file to the *Data/Road* folder of your Car-Maker project directory. Then, you can directly select the file in the file browser by clicking on the respective folder icon.

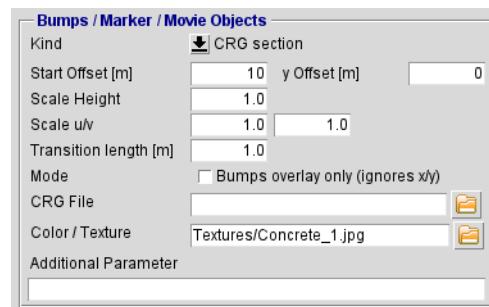


Figure 4.51: Integration of CRG road data using the IPGRoad dialog

Internally, the CRG data is parsed and the elevation profile is derived as a first step. As a second step, the surface (roughness profile) is mapped to a mesh. Therefore, the same interpolation methods as using the mesh bump are available.

It is possible to define multiple bumps and even multiple CRG-section which overlap each other.

The following is a list of all parameters which are used to parameterize the CRG section,:;

Setting	Description
Start Offset [m]	Distance between the beginning of the current road segment and the beginning of the CRG input.

Setting	Description
y Offset [m]	Lateral offset from the centerline (measured orthogonally to the direction vector). Note: if "bumps overlay" mode is activated the CRG profile may be distorted if used on turns.
Scale Height	Height information (z-values) from the .crg file will be multiplied by this factor.
Scale u	Length information (u-values) from the .crg file will be multiplied by this factor.
Scale v	Width information (v-values) from the .crg file will be multiplied by this factor.
Transition length [m]	The length of the transition between the normal road and the CRG section, which leads to a soft height adaption. Starts at the beginning of the CRG section.
Mode	If activated, only the height information is taken from the .crg file. If non-active, all road information including the longitudinal and lateral coordinates (x, y) are considered. Please note, that this might lead to a rotation of the road direction as a result of the tangential transition between the CRG data and the IPGRoad segments (see <a href="#">Figure 4.52</a> and <a href="#">Figure 4.54</a> ).
CRG File	Load your CRG data file from the Data/Road folder of your Car-Maker project directory.
Color/Texture	Insertion of: <ul style="list-style-type: none"> <li>RGB-colors and an optional alpha-channel with a range from 0-1.0 (e.g. 0.2,0.6,1.0,0.5)</li> <li>Textures from a file</li> </ul>
Additional Parameter	Additional Parameters to configure the CRG section. To edit the values, insert <i>KeyName=Value</i> . There is no space between the characters. Multiple values are divided by a blank between the different values. <ul style="list-style-type: none"> <li><i>imgX, imgY</i>: Length/Width of the texture in meters. Negative values mirror the texture</li> <li><i>angle</i>: direction angle in degree of the CRG section at beginning if "bumps overlay" mode is active</li> <li><i>tType</i>: defines the interpolation method 0: bicubic, 1: bilinear (default) 2: steps (sample &amp; hold)</li> <li><i>rampU0, rampU1</i>: defines the intersection length before and after the section in s-Road direction in meters (default=1m)</li> <li><i>rampV0, rampV1</i>: defines the intersection length to the right (V0) and left (V1) in y-direction of the road (default=1m)</li> <li><i>useElev</i>: Flag for usage of the derivated elevation profile. If set to 0, only the roughness from the mapped mesh is used</li> <li><i>useMesh</i>: Flag for usage of the mapped mesh. If set to 0, only the derivated elevation is used.</li> </ul>

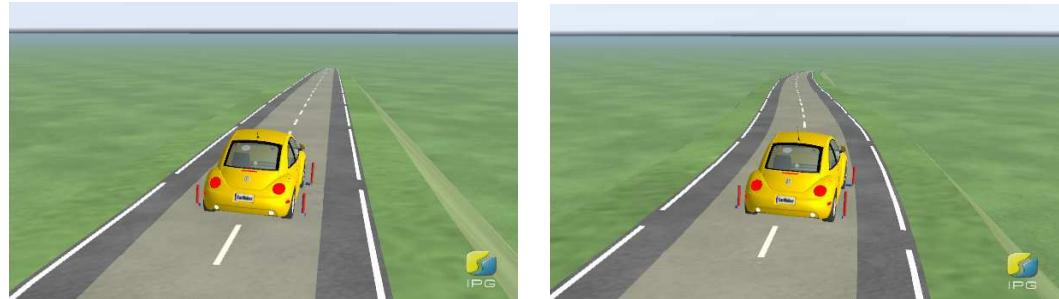


Figure 4.52: Difference in the *Mode* option: activated (= height information only, left), deactivated (= all road information, right)

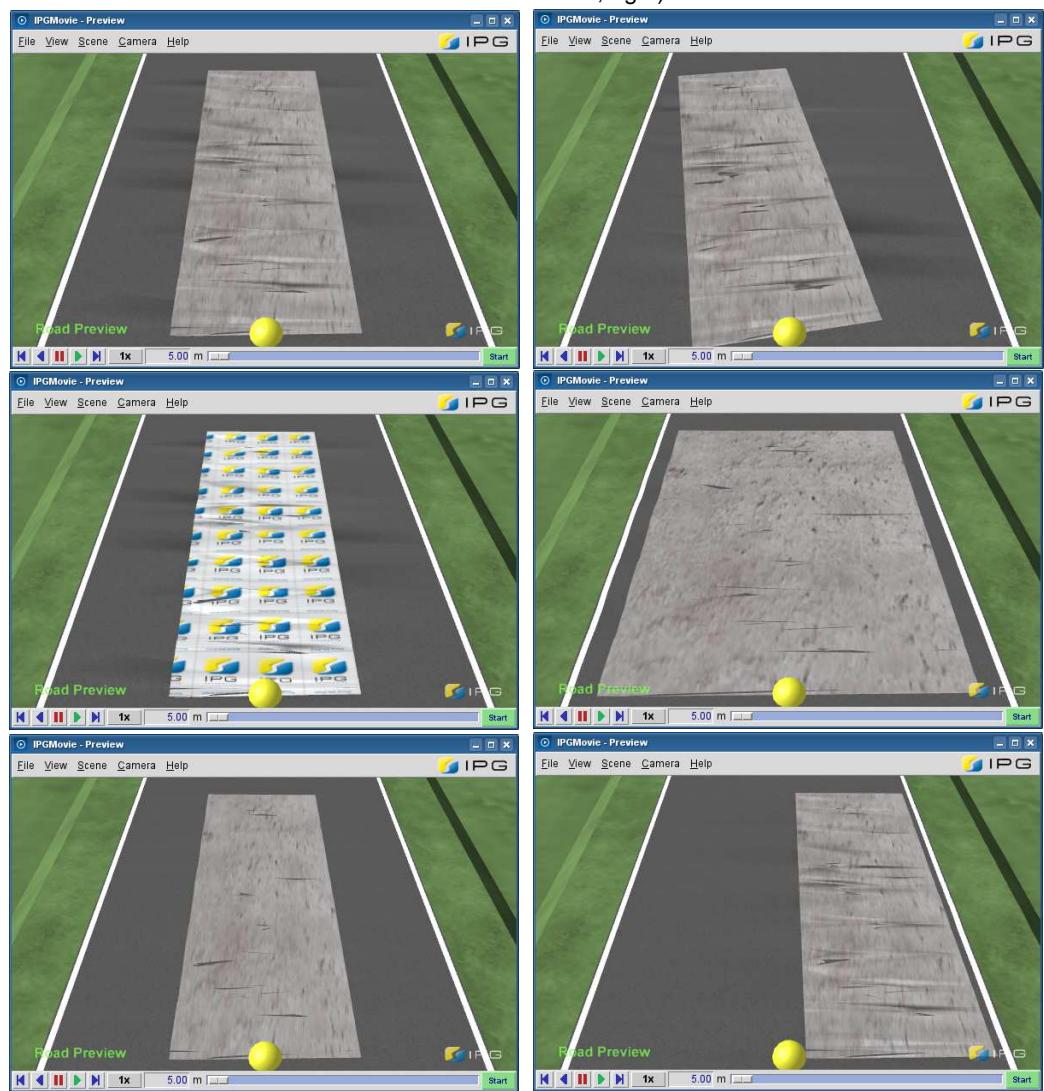


Figure 4.53: CRG section parameters changed: from top left to bottom right: default, angle=10, imgX/Y=1,scaleV=2,tType=0 and yOffset=-2



Please note the following restrictions in the usage of CRG road data:

- An initial height offset of the CRG data is ignored: the first point of the CRG grid is mapped to the height of the connected road segment from IPGRoad to avoid steps.

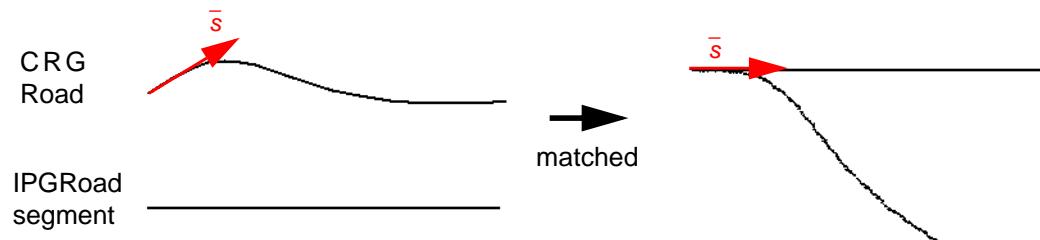


Figure 4.54: When *Mode* is deactivated, the direction vector of the CRG starting point is matched to the vector coming from the IPGRoad segment, where the CRG road data is inserted. This can lead to a rotation of the road orientation.

#### 4.4.9 Adding Sensor Objects

##### Road Marking

This marker enables you to draw marking lines on the road. The position as well as the shape and color of the lines are freely configurable. As some examples of use, this feature enables you to draw no-parking zones, pedestrian crossings, bicycle lanes and other useful markings on the road.



Figure 4.55: Insertion of road markings



Figure 4.56: Difference between straight and smoothed lines

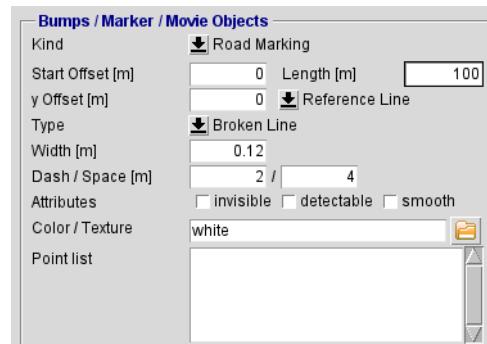


Figure 4.57: Parameterization of Road Markings

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
Length [m]	Total length of the road marking. If the value is -1, the road marking reaches until the end of the road.
y Offset [m]	Lateral offset from the reference (measured orthogonally to the direction vector). If the basis for the road marking is the reference line, a positive offset moves the road marking to the left. In case that the basis is a road border, a positive offset will result in a marking placed on the road.
Basis	On the arrow next to the y Offset, the basis for the road marking can be chosen. Actually the basis can be the reference line or the road borders.  Border left/right: creates a road marking relative to to left/right border Border both (2x): creates two road markings, one for each border Reference Line: creates a road marking relative to to reference line l+r of Reference Line (2x) : creates two road markings, one on the left and one on the right side of the reference line (only if y Offset > 0)
Type	Selectable types of the road marking: single line, double line, broken line, dotted line, left-broken double line, right broken double-line, left-dotted double line, right-dotted double line.
Width [m]	Total width of the road marking.
Dash / space [m]	Length and space for an interrupted line.
Attributes	<i>Invisible</i> : The road marking is invisible in IPGMovie but still considered by the line sensor (if detectable). <i>Detectable</i> : The Road Marking can be detected by line sensors. <i>Smooth</i> : The points are supporting points for a cubic spline.

Setting	Description
Color	<p>Color of the marking line (color name (black, white, red, green, blue, yellow, cyan, magenta, grey) or numerical RGB value with an optional alpha-channel with a range from 0-1.0 (e.g. 0.2, 0.6, 1.0, 0.5))</p> <p>Please note, that this color is independent of the color code that the Line Sensor detects (see <a href="#">section 5.23.6 'Line Sensor'</a>). The color code can be added in the <b>first</b> line of the point list using the following syntax: ColorIdx = &lt;code&gt; &lt;code&gt; must be an integer.</p>
Point list	Settable points for the marking line in X and Y coordinates. The point list have to start at 0 0. If it starts with different points, it will be supplemented automatically by 0 0.

**Traffic Barrier** This option places barriers on the road, e.g. to define roadways.

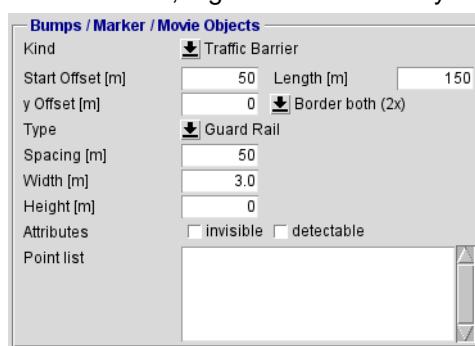


Figure 4.58: Parameterization of Traffic Barriers

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
Length [m]	Total length of the traffic barrier.
y Offset [m]	<p>Lateral offset from the reference line (measured orthogonally to the direction vector).</p> <p>If the basis for the traffic barrier is the reference line, a positive offset moves the road marking to the left. In case that the basis is a road border, a positive offset will result in a marking placed outside of the road.</p>
Basis	<p>On the arrow next to the y Offset, the basis for the road marking can be chosen. Actually the basis can be the reference line or the road borders.</p> <p><i>Border left/right</i>: Creates a traffic barrier relative to to left/right border</p> <p><i>Border both (2x)</i>: Creates two traffic barriers, one for each border</p> <p><i>Reference Line</i>: Creates a traffic barrier relative to to reference line</p> <p><i>l+r of Reference Line (2x)</i> : Creates two traffic barriers, one on the left and one on the right side of the reference line (only if y Offset &gt; 0)</p>

Setting	Description
Type	Different traffic barrier types are available. <i>Guard Rail</i> : single, double, triple) <i>Jersey Barrier</i> : continuous wall or single elements <i>Walls</i> with different textures: For the type wall, the vertical section of the start is normal to the surface of the road. For wall 2, the vertical section of the start is normal to the x-y-plane of the ground coordinate system. (see <a href="#">Figure 4.59</a> )
Spacing [m]	Length of one straight traffic barrier element (see <a href="#">Figure 4.60</a> )
Width [m]	Total width of the traffic barrier. (0: default)
Height [m]	Total height of the traffic barrier. (0: default)
Attributes	<i>Invisible</i> : The road marking is invisible in IPGMovie but still considered by the line sensor (if detectable). <i>Detectable</i> : The Road Marking can be detected by Line Sensors.
Point list	Settable points for the traffic barrier in X and Y coordinates. The point list have to start at 0 0. If it starts with different points, it will be supplemented automatically by 0 0 and is always smoothed. Additional parameters that can be set at this place: <i>MatA</i> sets the path to the barrier's texture (e.g. MatA=Textures/Fence_1.png) <i>MatAXY</i> defines the texture's scaling (e.g. <i>MatAXY</i> =2:4 means length in x direction is 2m and length in y direction is 4*2m)

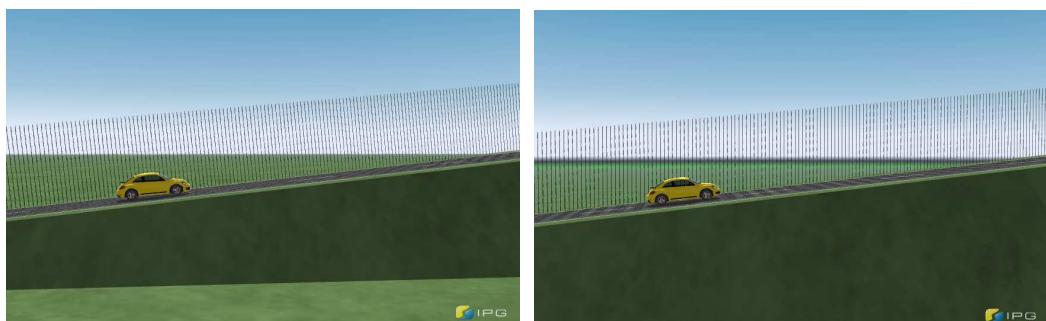


Figure 4.59: Traffic Barrier wall (left) and wall 2 (right)



Figure 4.60: Traffic Barrier big spacing (left) and small spacing (right)

**Traffic Signs** Using the traffic sign option, it is possible to insert road signs of different countries into your simulation environment. Signs of a specific country as well as a fallback dataset of another country can be selected via the *General Settings* tab. If the signs of the desired country are not available, you have the possibility to insert user defined signs.

The traffic sign files provided by IPG Automotive are located in the CarMaker installation directory within the subfolder *TrafficSigns*, the user defined signs can be saved in the project directory or data pool subfolder *TrafficSigns*. The subdirectory under *TrafficSigns* should be named using official country indicator according to ISO-Code 3166. If you need further information regarding user defined traffic signs, have a look at the Appendix B in the Reference Manual.



If the installation and project directory contain the same signs, the sign located in the project directory will be used.

The traffic signs can only be recognized by traffic sign sensors, not by IPGDriver. If the traffic sign option is activated, it is automatically valid for the whole TestRun.

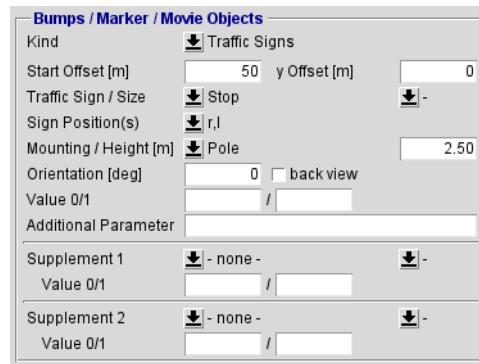


Figure 4.61: Parameterization of Traffic Signs

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
y Offset [m]	Lateral offset from the centerline (measured orthogonally to the direction vector).
Traffic Sign / Size	Different types of traffic signs and sizes which are predefined in the sign template.
Sign Position (s)	Selectable position of the sign: Left margin: relative to the left margin Right margin: relative to the right margin Reference Line: relative to the center Lane 0/1/2/3 left: relative to the center of the left lane Lane 0/1/2/3 right: relative to the center of the right lane
Mounting /Height [m]	The type of mounting (pole, double pole, frame on double pole, gantry, half-gantry) and height. When multiple traffic signs are placed on one position, the first sign sets the mounting type.
Orientation [deg]	Defines the orientation of the traffic sign. Activating the option "back view" rotates the traffic sign 180 degrees.
Value 0/1	Sets the values for the traffic sign. E.g. 50 for a Speed Limit sign.
Additional Parameter	At this point, parameters for expert users can be inserted.

Setting	Description
Supplement 1/2	At this point, you can select supplementary information, like a specific time or mass limit which will be displayed as an additional sign.

**Traffic Lights** Using this marker, it is possible to insert different types of traffic lights into your simulation environment.

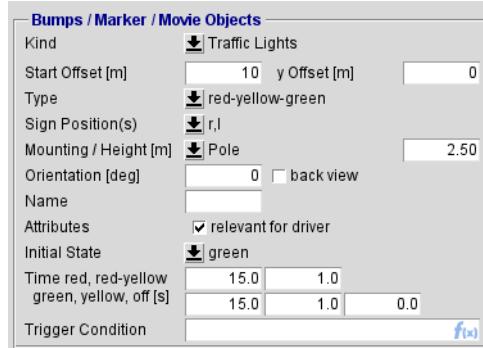


Figure 4.62: Parameterization of Traffic Lights

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
y Offset [m]	Lateral offset from the centerline (measured orthogonally to the direction vector).
Type	The type of the traffic light.
Sign Position (s)	Selectable lanes and margins for the position of the traffic lights.
Mounting/Height [m]	The type of mounting (pole, double pole, frame on double pole, gantry, half-gantry) and the height of the traffic light.
Orientation [deg]	The orientation of the traffic light definable in degrees. Activating the option "back view" rotates the traffic light 180 degrees.
Name	Freely definable name of the traffic lights. The name of the traffic light will also be the name of its quantity.
Attributes	Relevant for IPGDriver: When this tick box is selected, IPGDriver will consider the traffic lights during simulation.
Initial State	The initial state of the traffic lights: off, green, yellow, red, red-yellow.
Time red, red-yellow, green, yellow, off [s]	Freely definable times for each phase of the traffic lights.
Trigger Condition	At this point it is possible to trigger the change from the initial state of the traffic lights by using Realtime Expressions.

#### 4.4.10 Adding Movie Geometries

Movie geometries enhance the quality of the animation in terms of a beautification of the road and road environment. Please note that all objects inserted here will not have any effect on the simulation itself. They are ignored by IPGDriver and cannot be detected by DA Sensors.

To insert a new movie object proceed as explained in the beginning of [section 4.4.7 'Adding Road Markers' on page 58](#).

CarMaker offers the following types of movie geometries:

- Guide Post, to insert guide posts at the road side.
- Tree Strip, to picture alleys or single trees.
- Sign Plate, to display any individual type of sign like city name signs or traffic signs.
- Marking Line, to draw user defined lines on the road.
- Road Painting, to insert any texture on the road as a road painting.
- Geometry File, to insert an object beside the road.

**Guide Posts** This option enables to insert guide posts at the roadside. The following parameters are required:

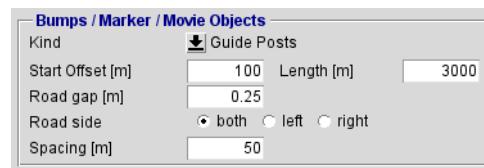


Figure 4.63: Parameterization of guide posts

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
Length [m]	Length on which the guide posts are displayed.
Road gap [m]	Lateral distance added to the track width.
Road side	Choose on which side the guide posts should be placed: on both sides or only on the left or right side.
Spacing [m]	Gap between two guide posts on one side.



Figure 4.64: Display of guide posts in IPGMovie



As an example take the TestRun Examples > CarMakerFunctions > IPGRoad > Road\_MovieObjects.

**Tree Strip(s)** To plant some trees beside the road, you can use the marker *Tree Strip(s)*. Trees can be placed on one or on both sides of the road. You can generate a whole wood or only some single trees to picture an alley. The following parameters are required:

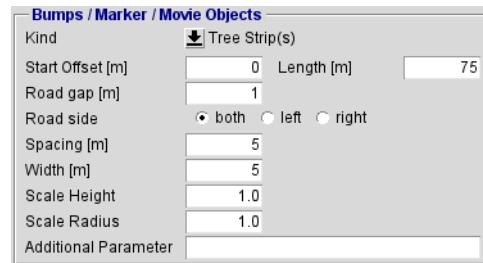


Figure 4.65: Parameterization of tree strips

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
Length [m]	Length on which the tree strip is displayed.
Road gap [m]	Lateral distance added to the track width.
Road side	Choose on which side the tree strips should be placed: on both sides or only on the left or right side.
Spacing [m]	Gap between two trees on one side.
Width [m]	Lateral extension of the tree strip on one side.
Scale Height	Scaling factor for the height of the trees.
Scale Radius	Scaling factor for the width of a single tree.
Additional Parameter	(optional) Texture applied to the geometry, see <a href="#">section 7.3.3 'Using Alternative Textures for Tree Strips'</a>



As an example take the TestRun Examples > CarMakerFunctions > IPGRoad > Road\_MovieObjects..



Figure 4.66: Visualization of tree strips in IPGMovie

**Sign Plate** This feature enables you to insert any graphical object on top of a pole, for example as a traffic sign. For the integration a texture drawing (\*.jpg, \*.png) is required which should be saved to the *Movie/Textures* folder of your CarMaker project directory. Following the same path within your installation directory, you will also find a number of objects which are included in the CarMaker package. These textures are available for all projects you are working in.

For the implementation of the texture the following values need to be defined:

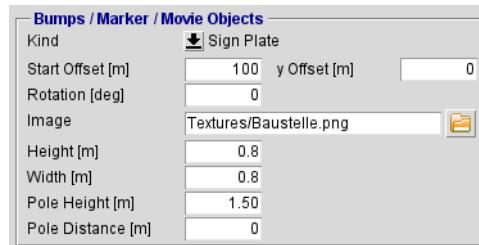


Figure 4.67: Parameterization of Sign Plate

Setting	Description
Start Offset [m]	Distance in meters from the origin of the respective segment (measured on the centerline).
y Offset [m]	Lateral offset from the centerline (measured orthogonally to the direction vector).
Rotation [deg]	Change the orientation of the plate. Positive angles lead to a clockwise rotation.
Image	Path to the texture, the default path already points to the <i>Movie</i> folder of the installation directory, e.g.: Textures/Sign_ZebraCrossing.png
Height [m]	Height of the sign plate.
Width [m]	Width of the sign plate.
Pole Height [m]	Vertical distance from the sign to the ground.
Pole Distance [m]	Distance between the poles. Enter 0 to generate only one pole.

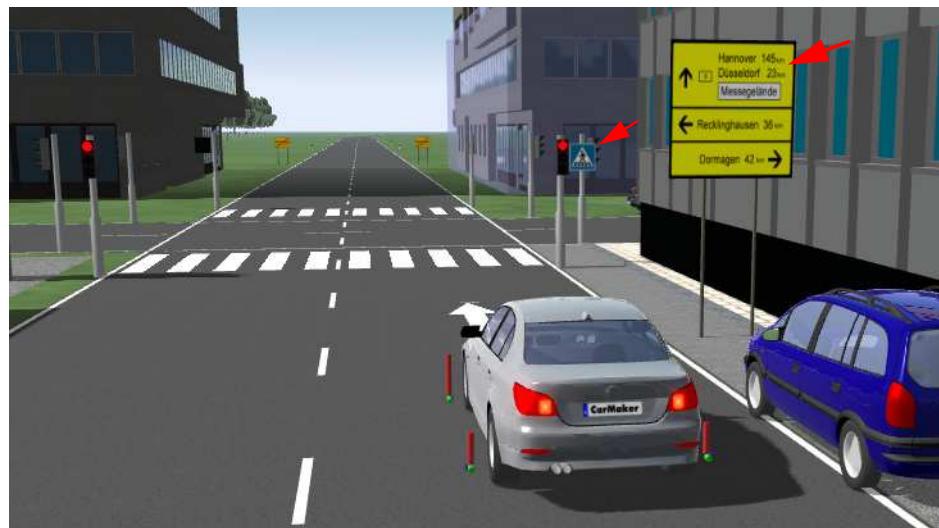


Figure 4.68: Insertion of traffic signs

**Road Painting** Using this marker, you are able to insert any texture drawing (\*.jpg, \*.png) as a road painting. The graphical object should be saved to the *Movie/Textures* folder of your current CarMaker project directory. CarMaker also provides a number of texture objects, which you can find under the same path inside your installation directory. The objects within this folder are available for all of your projects you are working in.

For the implementation of the texture the following quantities need to be defined:



Figure 4.69: Parameterization of road paintings

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
y Offset [m]	Lateral offset from the centerline (measured orthogonally to the direction vector).
Rotation [deg]	Change the orientation of the road painting. Positive angles lead to a clockwise rotation.
Image	Path to the texture, the default path already points to the <i>Movie</i> folder of the project directory.
Height [m]	Height of the road painting.
Width [m]	Width of the road painting.

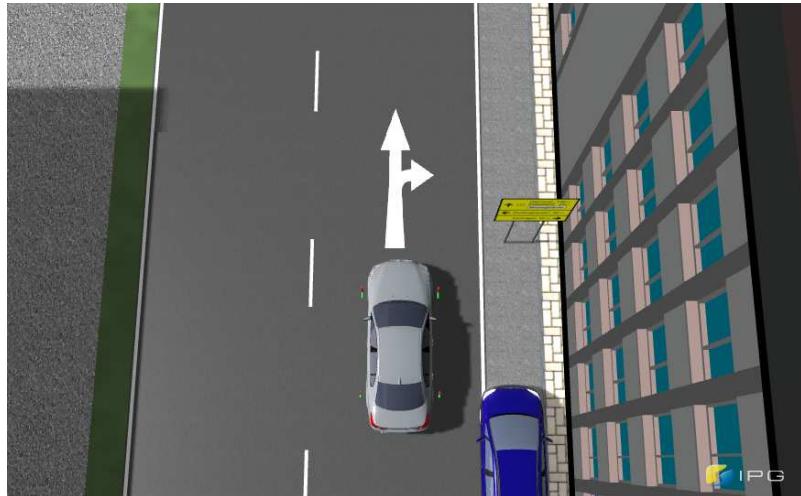


Figure 4.70: Insertion of road paintings



Instead of loading a \*.jpg or \*.png file, you can select a \*.mtex file, which describes the surface properties and is in the usual CarMaker infofile format. As an example, please check the file *Movie/Textures/Puddles/puddles03.mtex*. If you need further information regarding this topic, have a look at [section 7.4.2 'Integrating mtex-Files' on page 431](#).

**Movie Geometry** Using the movie geometry *Geometry File*, it is possible to display objects beside the road, which will have no effect on sensor systems.

For the implementation of the movie geometry the following quantities need to be defined:

Setting	Description
Start Offset [m]	Distance from the origin of the respective segment (measured on the centerline).
Length [m]	Total length of the movie geometry.
Road gap [m]	Lateral distance added to the track width.
Road side	Choose on which side of the road the movie geometry should be placed; on the right, the left or on both sides.
Spacing [m]	Gap between multiple movie geometries.
z Offset [m]	Height Offset of the movie geometry.
Rotation [deg]	Change the orientation of the movie geometry. Positive angles lead to a clockwise rotation.
Rel. to ground	Defines the orientation of the movie geometry around the z-axis. If rel. to ground is activated, the movie geometry is orthogonal to the coordinate system of the road surface instead of the coordinate system of the world.
Scale	Dimension of movie geometry will be multiplied by this factor.
Geometry File	Path to the geometry file.

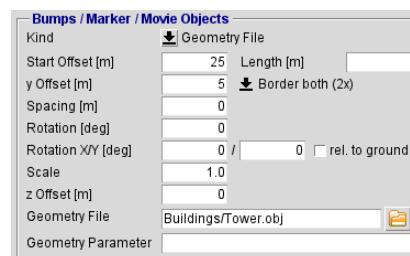


Figure 4.71: Definition of a geometry file in the Road dialog



Figure 4.72: Display of geometry files in IPGMovie

## 4.4.11 Special Road Configurations

In the *Road* dialog, you have the possibility to construct many different types of roads. Among other things, it is possible to build intersections, parking areas or highways including exits and returns.

In order to implement roads like these, you need to define the whole road as one coherent part. In the following section you will learn how to build complex and sophisticated road constructs with the help of some detailed examples.



When constructing the following special road configurations, make sure to disable the *Natural Slope* within the *IPGMovie Interface* to avoid the display of a green road margin stripe at those sections, which are containing overlay roads.

### Intersection



To build an intersection, IPGRoad offers you the possibility to build one single road which crosses itself at a predefined point. To implement these requirements, the road has to be constructed in the shape of a loop.

As an example take the *TestRun Examples > DriverAssistanceSensors > Autonomous-Functions > PedestrianDetection\_CrossstheRoad*.

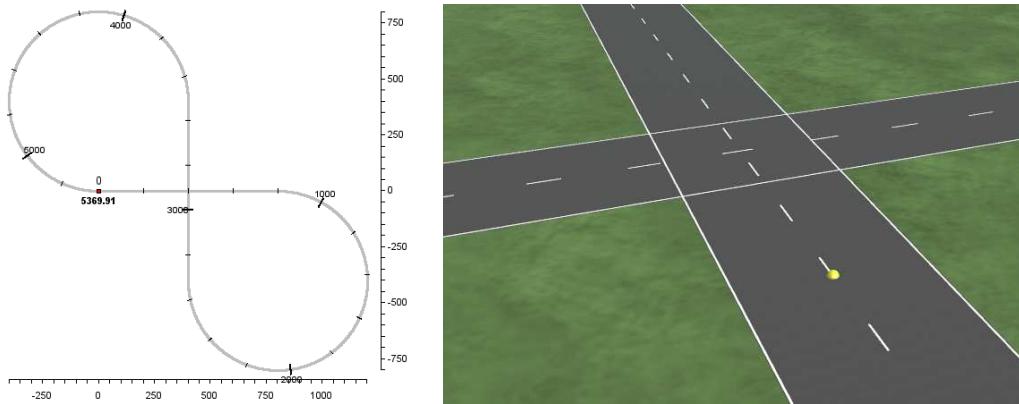


Figure 4.73: The view on an intersection

To build a road like shown above, you have to merge its segments in the following way:

#### Example

- 1st segment: straight, length 800 m
- 2nd segment: turn right, angle 270 deg, radius 400 m
- 3rd segment: straight, length 800 m
- 4th segment: turn left, angle 270 deg, radius 400 m

### Multi-Lane Roads



To build a multi-lane road, you have the possibility to build a road which turns around 180 degrees at its end and drives back parallel along the primary road.

As an example take the *TestRun Examples > CarMakerFunctions > IPGTraffic > Freeway-3lanes*.



Figure 4.74: The view on a multi lane road

#### Example

1st segment: turn left, angle 90 deg, radius 2500 m  
 2nd segment: turn left, angle 180 deg, radius 1.85 m  
 3rd segment: turn right, angle 89.9 deg, radius 2496.3 m  
 4th segment: turn left, angle 180 deg, radius 3.7 m  
 5th segment: turn left, angle 180 deg, radius 2503.7 m

An alternative is to build one wide road, which can be divided into different lanes with the help of marking lines. To get more information about this approach, have a look at the section *Road Painting* on page 71.

#### Highway Exit and Return

You have the possibility to build really complex highway scenarios including exits, returns and bridges with IPGRoad.



As an example take the TestRun Examples > DriverAssistanceSensors > ACCandPre-Crash > HighwayExitandReturn.

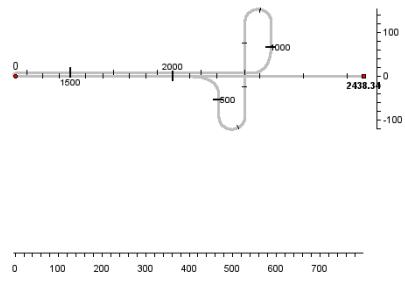


Figure 4.75: The view on a highway exit

To build a highway scenario like shown above, you can merge the road segments as follows:

**Example**

1st segment: straight, length 400 m  
2nd segment: clothoid right, angle 90 deg, radius 400 m, end radius 30 m  
3rd segment: straight, length 50 m, grad 2 %  
4th segment: turn left, angle 180 deg, radius 30 m, grad 2 %, slope 2 %  
5th segment: straight, length 216 m  
6th segment: turn right, angle 180 deg, radius 30 m, grad -2 %, slope -2 %  
7th segment: straight, length 50 m, grad -2 %  
8th segment: clothoid right, angle 90 deg, radius 400 m, end radius 30 m  
9th segment: straight, length 546 m  
10th segment: turn left, angle 180 deg, radius 3.98 m  
11th segment: straight, length 800 m

## 4.5 Maneuver

### 4.5.1 Overview

The concept of CarMaker for the driving scenario is the following:

- There is one maneuver definition, which is splitted into several maneuver steps (e.g: acceleration, braking, ...). These maneuver steps are called *minimaneuvers*. Each minimaneuver is composed of
  - longitudinal dynamic actions: accelerating, braking, gear shifting, ...
  - lateral dynamic actions: steering,
  - additional actions, defined by a list of special minimaneuver commands of a very easy script language.
- Please note the following features:
  - CarMaker offers the possibility to use real measurements as input (see [section 4.6 'Input From File / Using Real Measurements' on page 110](#)).
  - Starting a maneuver with an initial speed is possible: see [section 4.5.5 'Special Maneuvers - Definition of start and end conditions' on page 98](#).
  - Logical expressions evaluated in realtime can be implemented. Fields of applications of the so called *RealtimeExpressions* are the maneuver start conditions (see [section 'Trigger' on page 108](#)) and the Minimaneuver Command Language (see [section 4.5.8 'Minimaneuver Command Language' on page 93](#)).

The first thing is to define the various minimaneuvers you need. There are several ways to configure the driving maneuvers and the driving behavior:

- in the TestRun (TestRun specific),
- in the vehicle model parameter set (vehicle specific),
- in the SimParameters parameter set (global).

At first we will learn how to define a maneuver scenario in a TestRun.



Please note that reading section "Defining the Maneuver" in the Quick Start Guide to have the necessary background is a requirement before going further.

## 4.5.2 Building a Scenario - Creating a Maneuver list

CarMaker offers a convenient GUI to build a great diversity of driving scenarios. Each driving scenario enables to control the vehicle, e.g. to control the steering wheel and the vehicle pedals.

This interface is limited to the definition of a driving scenario with only one vehicle. If you wish to simulate the vehicle in a traffic, please read the section [4.8 'Traffic' on page 127](#).

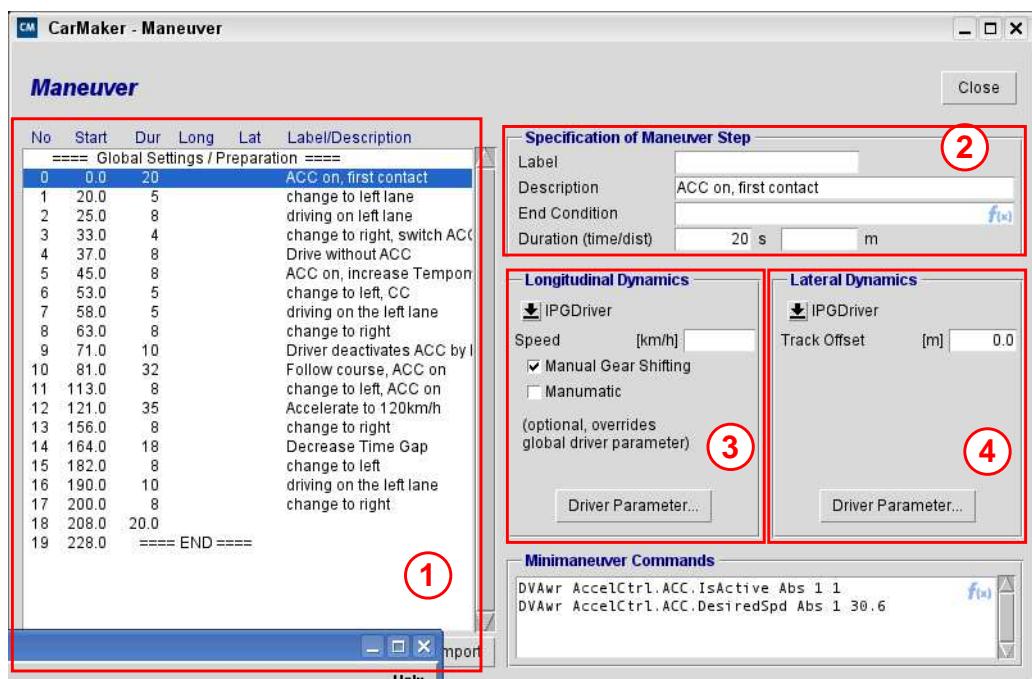


Figure 4.76: The Maneuver dialog

To build a driving scenario, you have to add and parameterize successively the various maneuver steps necessary to control the vehicle as you want.

- Maneuver List:**
- New**
- Copy**
- Insert**
- Delete**
- Import**

The list of the maneuver steps is displayed and can be built in box 1 of [Figure 4.76](#) at the left side of the Maneuver dialog.

Here the maneuver steps are added or deleted. A new maneuver step is added by using the *New* button at the bottom of box 1. The maneuver steps are always inserted ahead of the one selected. A maneuver step can be selected by clicking on it in the maneuver list in box 1 - it is then highlighted in blue. To add a step to the end of the list, click on the entry "==== END ====" to select it, and then click on the button *New*.

Maneuver steps can also be copied and pasted, deleted or imported from another TestRun. You can find the same functionality *Import* by clicking with the right mouse button anywhere in the Maneuver GUI. In this case you will find also an option *Clear*, used to delete all steps at once.

After creating a maneuver step, the actions need to be defined. Each minimaneuver consists of a duration (box 2 in [Figure 4.76](#)) and a description of the driver's task, separated in longitudinal and lateral dynamics (box 3+4 in [Figure 4.76](#)).

### 4.5.3 Specification of a Maneuver Step

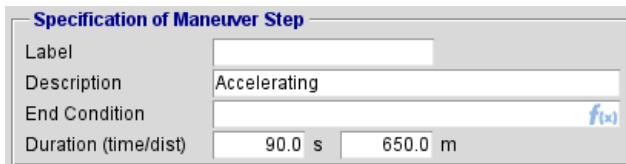


Figure 4.77: Box 2: Specification of a maneuver step

- Label** Optional parameter. You can specify a label for the maneuver step. The string entered in this field can be used to address this minimaneuver, e.g. when jumping from one maneuver step to another. Please find more information about maneuver jumps in [section B.1.1 'Driving Maneuver Commands' on page 590](#).
- Description** Optional parameter. This is a comment to the maneuver step. This comment is also displayed in the maneuver list in the CarMaker main GUI.

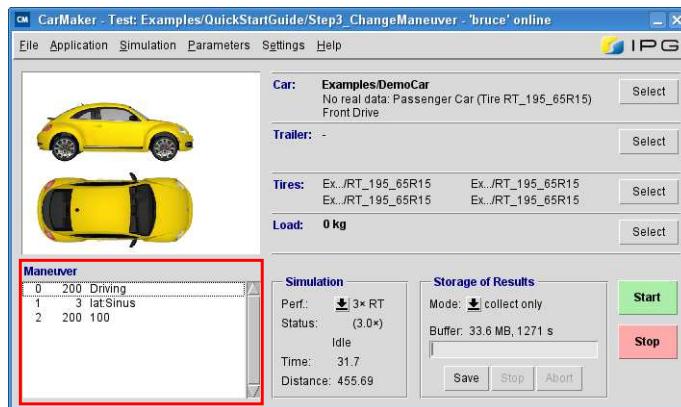


Figure 4.78: Maneuver description displayed in the CarMaker main GUI

- End Condition** Each maneuver step needs to have an end condition. The maneuver interface offers several possibilities to end a maneuver step. All possibilities are listed hereunder:

Table 4.4: Maneuver End Conditions

Kind of Condition	Description
End Condition	User defined end condition using Realtime Expressions (see section 'Realtime Expressions' on page 602).
Time	A duration defines the end of the step. When the time reaches this value, the step ends and the scenario continues on the next step. Further explanations can be found in the paragraphs below.
Distance	A distance defines the end of the step. When the vehicle reaches this distance, the step ends and the scenario continues on the next step. Further explanations can be found in the paragraphs below.
DrivMan jump	With the Minimaneuver Command Language (see section 'Minimaneuver Command Language' on page 590), you have the possibility to jump to another maneuver step according to various conditions. You can use a similar commands with ScriptControl (Power User only!).

Table 4.4: Maneuver End Conditions

Kind of Condition	Description
Road Marker	At a given point on the road, you can trigger the end of the current maneuver step, and jump to another maneuver step. See the section B.1.1 'Driving Maneuver Commands' on page 590.

**User defined End Condition** In the field *End Condition* you can specify any end condition you like. With a click on the right mouse button in this field you have access to all Data Directory quantities of CarMaker. Select one and combine it with a logical operator to create a condition. The condition is evaluated each simulation cycle. Once it becomes true, the simulation stops. For example, if the simulation should be finished after the car reaches a velocity of 100 kph, you would select the quantity Car.v from the menu of the right mouse and enter:

`Car.v >= 100/3.6`

Please note that Realtime Expressions always use SI units.



Specification of Maneuver Step	
Label	
Description	Accelerating
End Condition	<code>Car.v&gt;=100/3.6 &amp;&amp; KI15SW==1</code> <a href="#">f(x)</a>
Duration (time/dist)	90.0 s    650.0 m

Figure 4.79: Using Realtime Expressions to define a dynamic end condition

Please find a list of all possible operators and functions in [section 'Realtime Expressions' on page 602](#).



As an example take the TestRun Examples > CarMakerFunctions > RTEexpressions > DynamicEndConditions.

**Duration (s) or (m)** The classical end conditions - time or distance - can be directly entered in the fields *Duration*.

Specification of Maneuver Step	
Label	
Description	Accelerating
End Condition	
Duration (time/dist)	90.0 s    650.0 m

Figure 4.80: Defining the duration of a maneuver step

Both time and distance count from the beginning of the maneuver step and do not apply to the overall passed time or distance in the whole simulation (see UAQs *DM.ManTime* and *DM.ManDist*).

If you specify a value in both fields, time AND distance, then the condition which is fulfilled at first is used as end condition of the maneuver step.

For instance:

- Let us suppose the first maneuver is accelerating on a straight line. The maneuver duration is set to a time of 2 s and a distance of 200 m. Obviously the vehicle will not accelerate fast enough to reach 200 m within 2 s. The maneuver step ends therefore at 2 s, and the driving scenario will continue with the maneuver step that comes after.

## 4.5.4 Global Settings / Preparation

The first entry in the list of maneuver steps has a special meaning. It represents an user interface for special maneuver settings.

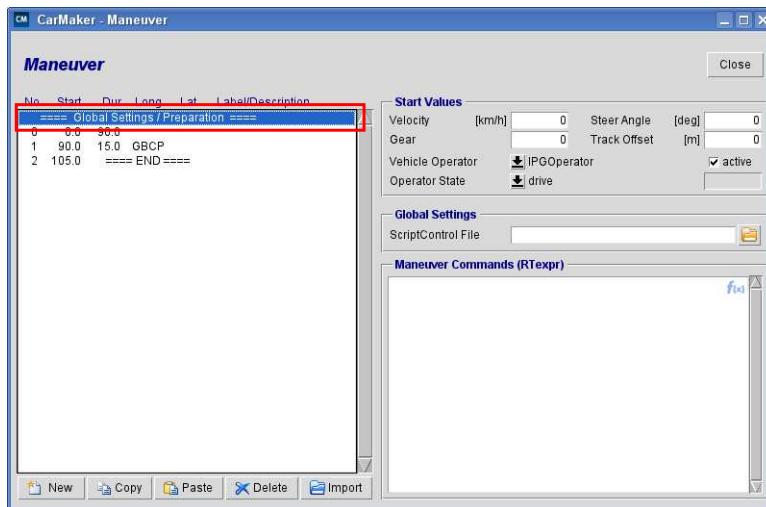


Figure 4.81: Section Global Settings / Preparation of the Maneuver dialog

### ScriptControl File

In this part, a ScriptControl script file can be loaded. However, the commands the script file contains are not executed automatically. The loaded script file rather serves as function library. To call the procedures use the *Realtime Expression* command *signal* (see [signal\(\)](#)). Please find further information on the programming language ScriptControl in the Programmer's Guide.

### Maneuver Commands (RTexpr)

In this section you can place any kind of *Realtime Expression* commands (see [section C.2 'Operators and Functions' on page 603](#)). The Realtime Expressions are executed during the entire TestRun. As opposed to Realtime Expressions defined under *Minimaneuver Commands* (see [section 4.5.8 'Minimaneuver Command Language' on page 107](#)) in a single maneuver step, the commands entered here are valid for the whole TestRun. They are not bound to a certain maneuver step.



Another difference is the syntax: The *Minimaneuver Commands* window expects a prefix called "Eval", to indicate the command as a Realtime Expression. This is not needed here.

#### Example

The following command defined under *Global Settings* will observe the lateral acceleration throughout the entire TestRun:

**Car.ay>3.0 ? MyFlag=1**

#### Example

The same command in the *Minimaneuver Commands* window needs a prefix and will be valid only in that minimaneuver it was defined in:

**Eval Car.ay>3.0 ? MyFlag=1**

### Start Values

The first entry in the maneuver list mainly serves to specify starting conditions. Possible settings are the initial speed, the gear number, the steering wheel angle and track offset.



As an example take the TestRun Examples > CarMakerFunctions > ManeuverControl > SetStartingSpeed.

**Vehicle Operator** The *Vehicle Operator* simulates the vehicle operation state at the beginning of the simulation. There are application cases, which require to start the vehicle in other states than with engine on and ready to drive. The following operator states are available:

Table 4.5: Operator states of IPG Vehicle Operator

State	Description
absent	Driver has left the vehicle
poweroff	Driver is in the vehicle, key is inserted, but power is still off
poweracc	Power accessory available, e.g. for car radio etc.
poweron	Power is switched on
drive	Ignition and engine on, vehicle is ready to drive
user defined <i>	User defined state, sends user defined signal from vehicle operator to powertrain control (e.g. seat belt, vehicle door, seat sensor), using UAQs DM.UserSignal_<i>

The operation state machine itself is defined in the PTControl Unit, see [section 5.20.1 'Parametrizing PT Control' on page 257](#).

Each state influences certain signals relevant for the vehicle operation. For the IPGOperator, the settings are as follows (please find further information about the meaning of the quantities (UAQ) in the Reference Manual, chapter "User Accessible Quantities"):

Table 4.6: Start configuration for operator mode *absent*

UAQ	Value
DM.Key	0: key out
DM.GearNo	0
DM.SelectorCtrl	SelectorCtrl_P
DM.Gas	0.0
DM.Clutch	0.0
DM.Brake	0.0
DM.BrakePark	1.0
DM.BrakeLever	0.0
DM.SteerAng	0.0
DM.SteerTrq	0.0

Table 4.7: Start configuration for operator mode *poweroff*

UAQ	Value
DM.Key	1: key in, power off
DM.GearNo	0
DM.SelectorCtrl	SelectorCtrl_P
DM.Gas	0.0
DM.Clutch	0.0
DM.Brake	0.0

Table 4.7: Start configuration for operator mode *poweroff*

UAQ	Value
DM.BrakePark	1.0
DM.BrakeLever	0.0
DM.SteerAng	0.0
DM.SteerTrq	0.0

Table 4.8: Start configuration for operator mode *poweracc*

UAQ	Value
DM.Key	2: key in, power accessory
DM.GearNo	0
DM.SelectorCtrl	SelectorCtrl_P
DM.Gas	0.0
DM.Clutch	0.0
DM.Brake	0.0
DM.BrakePark	1.0
DM.BrakeLever	0.0
DM.SteerAng	not set
DM.SteerTrq	0.0

Table 4.9: Start configuration for operator mode *poweron*

UAQ	Value
DM.Key	3: key in, power on
DM.GearNo	0
DM.SelectorCtrl	SelectorCtrl_N
DM.Gas	0.0
DM.Clutch	1.0
DM.Brake	1.0
DM.BrakePark	0.0
DM.BrakeLever	1.0
DM.SteerAng	not set
DM.SteerTrq	0.0

Table 4.10: Start configuration for operator mode *drive*

UAQ	Value
DM.Key	4: driving
DM.GearNo	1
DM.SelectorCtrl	SelectorCtrl_D
DM.Gas	0.0
DM.Clutch	1.0
DM.Brake	0.0

Table 4.10: Start configuration for operator mode *drive*

UAQ	Value
DM.BrakePark	0.0
DM.BrakeLever	0.0
DM.SteerAng	not set
DM.SteerTrq	0.0

## 4.5.5 Special Maneuvers - Definition of start and end conditions

There are several ways to define start conditions in a TestRun. Start conditions can be used to set an initial speed or a steer offset right at the beginning.

- Maneuver dialog

As explained in the section above, some start values like velocity, gear number, steering wheel angle and track offset can be defined in the first entry *Global Settings / Preparation* of the Maneuver dialog.

Please note that these settings are applied at the beginning of the TestRun - the vehicle is given the initial speed but it needs to settle and overcome powertrain inertias etc. first. This can lead to a non-constant velocity signal at the first seconds of the simulation.

- Input From File

The *Input From File* mechanism is another possibility to specify starting conditions. A start velocity, gear number and throttle position can be specified. As opposed to the starting conditions in the Maneuver dialog, in this case the vehicle is accelerated in the preparation phase of the simulation. Thus, you will observe a driving car before the simulation starts - but then the vehicle is already settled at the given initial conditions which prevents disturbances in the signal sequence.

Please find more information about *Input From File* in [section 4.6 'Input From File / Using Real Measurements' on page 110](#).

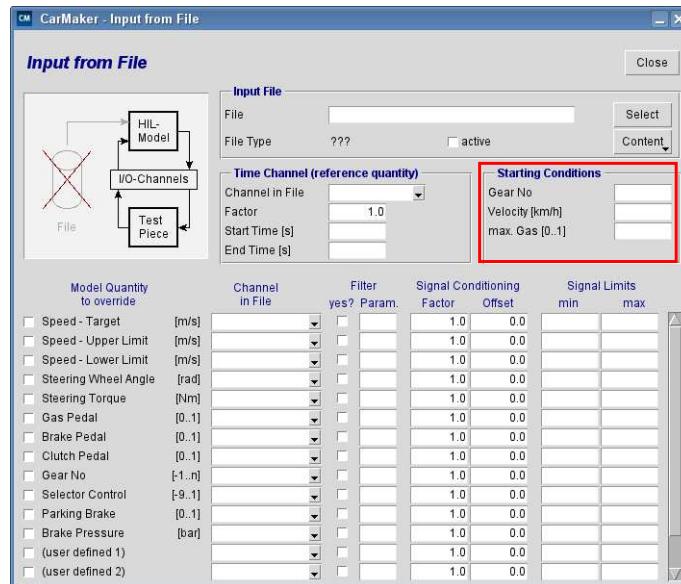


Figure 4.82: Specifying starting conditions in the Input From File dialog

- Determine Start Values

With this feature a snapshot is taken at a pre-defined moment during the simulation to save the current vehicle state.

In the CarMaker GUI go to Simulation > Determine Start Values to take a snapshot of the vehicle state at an arbitrary time or distance during a TestRun.

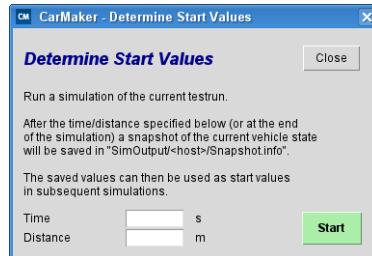


Figure 4.83: Determine start values

CarMaker will store the current vehicle state into the SimOutput/<hostname>/Snapshot.info file. These values can be used to setup the vehicle's start conditions. To do so, you just have to edit the generated snapshot file (ASCII file) and copy/paste all the parameters in a given vehicle Infofile (either by editing the Infofile directly with your habitual text editor or by using the *Vehicle Data Set Editor* and pasting the parameters in the field *Additional Parameters*, tab *Misc.*, see [section 5.25 'Miscellaneous' on page 299](#)).

For further details about this functionality, check the Reference Manual, [section 'Start Conditions'](#).

- Final Maneuver

If the last minimaneuver has a duration of zero seconds, it has a special meaning:

This maneuver is called at the end of the TestRun even if the TestRun was stopped ahead of time (by user, by an error, ...). The actions defined here will be conducted at any rate.

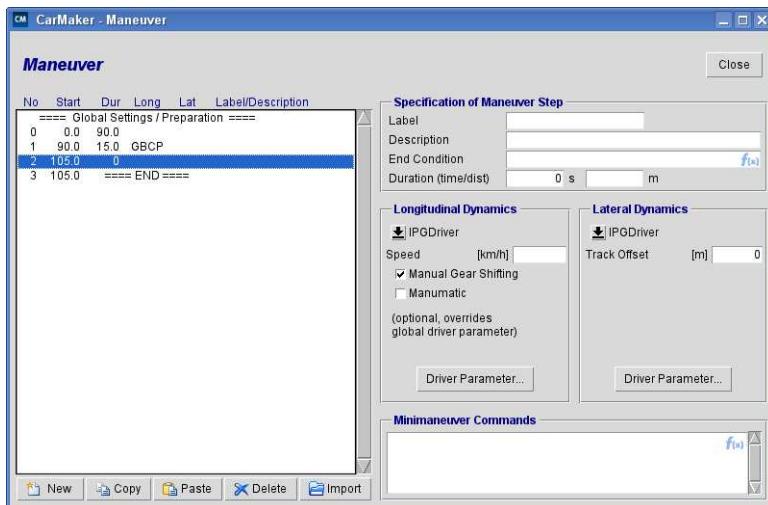


Figure 4.84: Final Maneuver with duration zero

This special final maneuver step is a very convenient way to e.g. reset quantities which were overwritten by DVA commands ([section 'DVArcl - DVA release' on page 594](#)) during the TestRun or to trigger post processing routines in combination with user procedures defined in a tcl script (see [section 'Start Values'](#)).

As an example take the TestRun Examples > CarMakerFunctions > ManeuverControl > CleanupAtEnd.



## 4.5.6 Longitudinal Dynamics

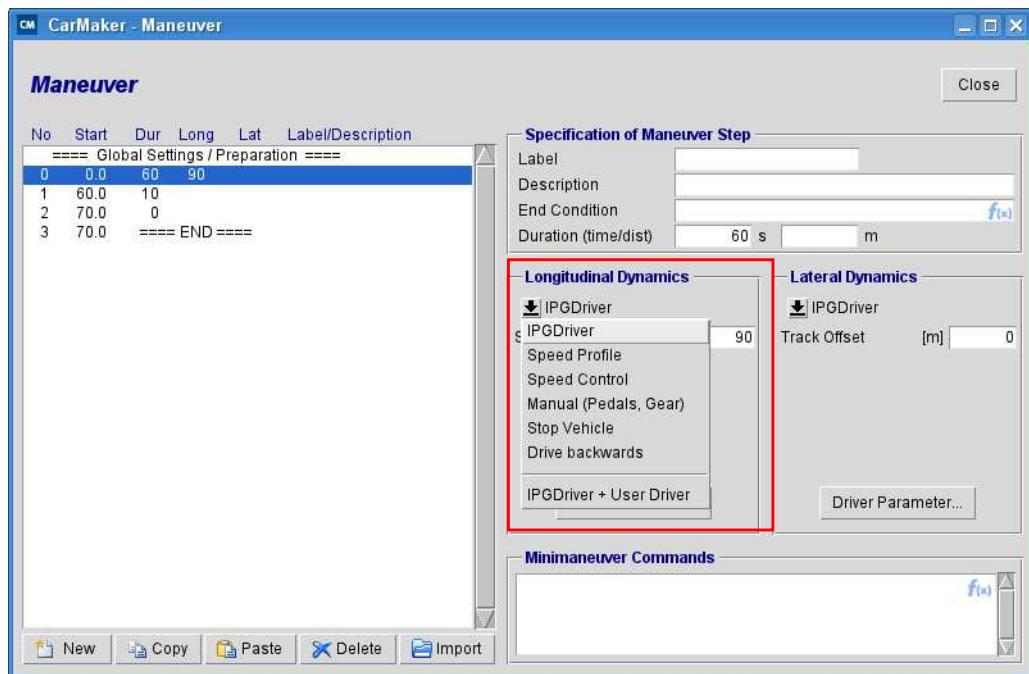


Figure 4.85: Possible specifications of longitudinal dynamics

The following options are available to control the pedals' positions:

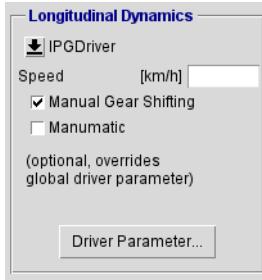
Table 4.11: Maneuver Step Longitudinal Dynamics Options

Option	Description
IPGDriver	Fully automated and detailed speed and pedal control. Note that you can also activate the driver model for the lateral dynamics in order to let him control the whole vehicle (see next subsection).
Speed Profile	Follow measured speed read from an external file.
Speed Control	Simple speed controller.
Manual	Manual control of the pedals' positions.
Stop Vehicle	Stop the vehicle. Based on IPGDriver.
Drive Backwards	Drive backwards. Based on IPGDriver.
User Driver	Overwrite certain attributes of IPGDriver by a user defined driver model.

Let us give some hints to help you to choose the best option for your maneuver steps.

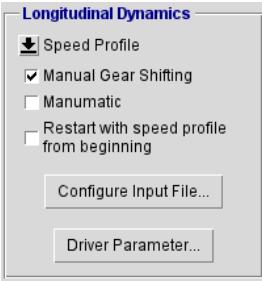
### IPGDriver

To accelerate and brake the vehicle according to its dynamic limits, the option IPGDriver should be chosen (closed loop maneuver). This option activates the driver model for the longitudinal dynamics.

<b>Speed (km/h)</b>	Optional parameter. Defines the speed the driver tries to reach on the road. This parameter is optional and depends on each maneuver step.  The default value for the <i>Speed</i> applies for all maneuver steps, and is specified in the IPGDriver dialog (see <i>Driver Parameter...</i> ).	
<b>Manual Gear Shifting</b>	If a vehicle with a manual gear box is used in this TestRun, this option should be activated. It tells IPGDriver to shift according to the shifting strategy defined in the IPGDriver dialog (see <a href="#">section 4.7.2 'User parameterized Driver' on page 118</a> ). Is this option deactivated, IPGDriver will not shift but stick in the gear used in the previous maneuver step.	
<b>Manumatic</b>	In case a vehicle with automatic gear box is used, IPGDriver can overrule the shifting logic of the ECU. For this, activate the manumatic mode. However, the manumatic mode needs to be activated in the TCU (see <a href="#">section 5.20.3 'Parametrizing the TCU' on page 268</a> ).	
<b>Driver Parameter...</b>	Clicking on this button opens the GUI in which the driver model can be parameterized. See the <a href="#">section 4.7 'Driver' on page 116</a> to learn how to do it.	

## Speed Profile

The option *Speed Profile* is relevant only if you have some vehicle speed measurements. It is in fact a simple driver model (IPGDriver Lite), that follows the defined speed measurements.

<b>Manual Gear Shifting</b>	When using measurements from an external file, you can choose to let CarMaker shift the gear box if the gear number is not registered in the file.  In such a case, this option should be activated. The range of the engine speed has to be parameterized in the IPGDriver dialog (see <a href="#">section 4.7.2 'User parameterized Driver' on page 118</a> ).	
<b>Manumatic</b>	In case a vehicle with automatic gear box is used, IPGDriver can overrule the shifting logic of the gearbox ECU. For this, activate the manumatic mode. However, the manumatic mode needs to be activated in the TCU (see <a href="#">section 5.20.3 'Parametrizing the TCU' on page 268</a> ).	
<b>Restart with speed profile from beginning</b>	This option enables to read the file again when the maneuver step starts in which the option is activated.	
<b>Configure Input File...</b>	Clicking on this button opens the GUI in which the file where the measurements are stored can be selected. See the <a href="#">section 4.6 'Input From File / Using Real Measurements' on page 110</a> to learn how to do it.	
<b>Driver Parameter...</b>	Clicking on this button opens the GUI in which the driver model can be parameterized. See the <a href="#">section 4.7 'Driver' on page 116</a> to learn how to do it.	

## Speed Control

The option Speed Control activates a simple speed controller. Contrary to the IPGDriver, the Speed Control option can not brake the vehicle: once the specified speed is reached, it will be kept without considering the curves of the road.

However, it can be useful if you do not have too high accuracy requirements and if you do not need to brake the vehicle. It has also the advantage that fewer parameters need to be defined than in IPGDriver (three parameters instead of nine, at the minimum!).

Longitudinal Dynamics		
<input checked="" type="checkbox"/>	Speed Control	
Speed	[km/h]	50
Max. Devition	[km/h]	0.0
Sensitivity	[0..1]	1.0
<input checked="" type="checkbox"/> Manual Gear Shifting		
<input type="checkbox"/> Manumatic		
<input type="checkbox"/> Premature end when final speed is reached		

**Speed (km/h)** The desired vehicle speed has to be specified. The controller will accelerate to or keep this speed according to a PID-controller.

**Max. Deviation (-)** The maximal deviation of the desired speed defined above has to be specified here.

**Sensitivity (-)** Defines the gradient of the control parameter: it determines how fast the desired speed should be reached. By this factor the controller exit is multiplied with. In general, it is small at a larger deviation.

**Manual Gear Shifting** If a vehicle with a manual gear box is used in this TestRun, this option should be activated. It tells IPGDriver to shift according to the shifting strategy defined in the IPGDriver dialog (see [section 4.7.2 'User parameterized Driver' on page 118](#)). Is this option deactivated, IPGDriver will not shift but stick in the gear used in the previous maneuver step.

**Manumatic** In case a vehicle with automatic gear box is used, IPGDriver can overrule the shifting logic of the gearbox ECU. For this, activate the manumatic mode. However, the manumatic mode needs to be activated in the TCU (see [section 5.20.3 'Parametrizing the TCU' on page 268](#)).

**Premature end when final speed is reached** This option aborts this maneuver step independently of the defined end conditions, if the target speed is reached.

## Manual

The option Manual can be chosen to control the position of the pedals with great accuracy. Of course, this option is much more time consuming to define than any other maneuver step, but it allows you to act accurately on the pedals at the time you want.

Longitudinal Dynamics			
<input checked="" type="checkbox"/>	Manual (Pedals, Gear)		
Clutch	Value	Start	dt
Gas	0	0.2	<input type="checkbox"/>
Brake	0	0.2	<input type="checkbox"/>
BrakePark	0	0.2	<input type="checkbox"/>
Gear	0	0.0	<input type="checkbox"/>
value = 0..1; +/- = value is offset			

**Clutch, Gas, Brake, BrakePark, Gear (-)** Each of those terms refers to the vehicle pedals or gear. For each of them, the following properties have to be defined:

- Value

It is the absolute value of the pedal position, from 0 to 1. The value 0 means pedal released, 1 means pedal fully pressed.

- Start

It specifies from which moment on the defined value has to be applied. The time in seconds starts counting at the beginning of the maneuver step.

- dt

It specifies the time required in seconds to reach the defined position.

- +/-

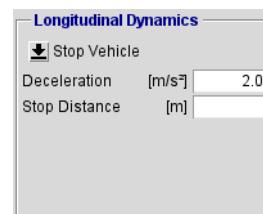
By activating this option, you can choose to define the value as offset to the former maneuver step, and not as absolute value (as by default).

As an example take the TestRun Examples > VehicleDynamics > Braking.



### Stop Vehicle

The *Stop Vehicle* option should be chosen if you do not need to control accurately the pedal position to stop the vehicle. If both of the following values are specified, the highest one is valid.

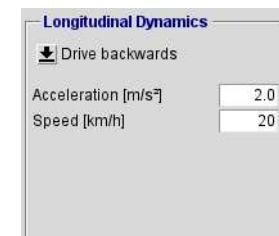


**Deceleration (m/s<sup>2</sup>)** Specifies a deceleration IPGDriver stops the vehicle with, from his current speed up to stand still. The deceleration value must be positive.

**Stop Distance** Specifies the distance, which the vehicle need to up to stand still.

### Drive Backwards

The *Drive Backwards* option should be chosen if you do not need to control accurately the pedal position to drive the vehicle backwards. The vehicle should be standing still before the beginning of this maneuver.



**Speed (km/h)** It specifies which speed IPGDriver should reach. This speed must be positive.

**Acceleration (m/s<sup>2</sup>)** It specifies the acceleration IPGDriver reaches, from stand still up to the speed defined above. The Acceleration value must be positive.

## 4.5.7 Lateral Dynamics

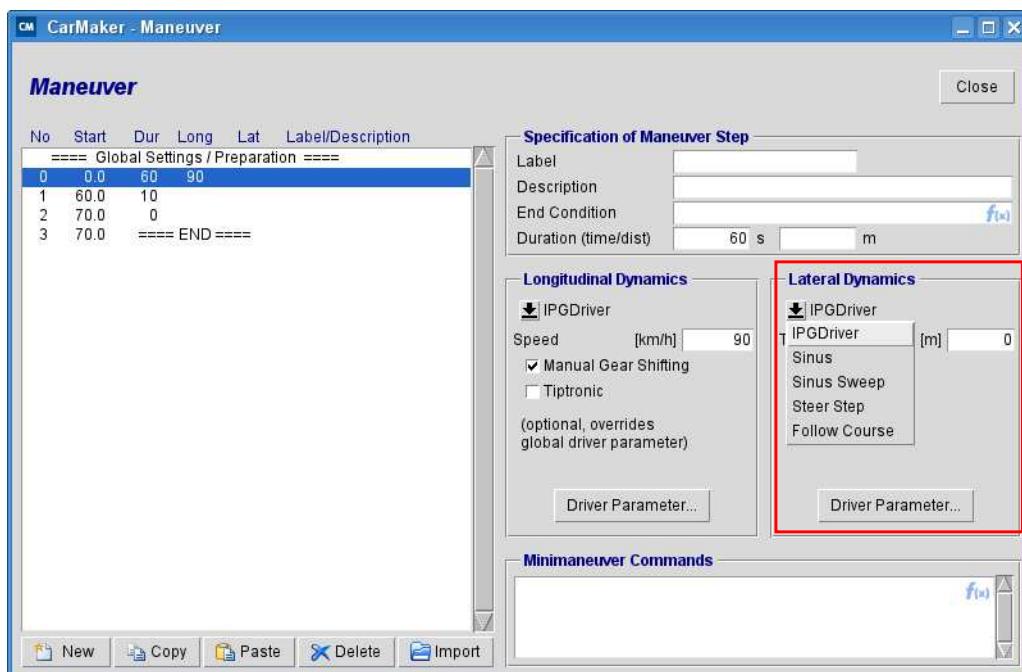


Figure 4.86: Possible specifications of lateral dynamics

Four options are available to control the steering wheel:

Table 4.12: Maneuver Step Longitudinal Dynamics Options

Option	Description
IPGDriver	Fully automated and refined steering wheel control. Note that you can also activate the driver model for the longitudinal dynamics in order to let him control the whole vehicle (see previous subsection).
Sinus	Sinus steering input.
Sinus Sweep	Ascending sinus steering input.
Steer Step	Steer step input.
Follow course	Simple steering wheel controller.

Let us give some hints to help you choose the best option for your maneuver steps.

### IPGDriver

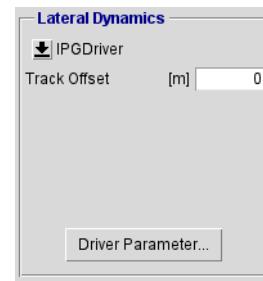
To steer the vehicle according to its dynamics limits, the option IPGDriver should be chosen. This option activates the driver model for the lateral dynamics.

#### Track Offset (m)

Here you can define the driver's deviation from its static desired course, e.g. to overtake a vehicle or to avoid a road obstacle. By default, leave this value to 0.0. A positive value moves the vehicle to the left, and a negative one to the right.



Note that when you define a value different to the value in the previous maneuver step, then the new value will be built up steadily throughout the entire step duration. This means if a step that lasts 15 s is defined with a Track Offset of 4 m, then the driver will take the entire 15 s to move the vehicle 4 m to the left.



For the definition of the global driving lane the driver uses please refer to the RoadGUI, Global Settings, [chapter 4.4, section 'Driving Lane' on page 42](#).



Pay also attention to the following point: If you wish to keep a track offset during several maneuver steps, do not forget to define the same value for each of them. Indeed, the offset is always calculated compared to the driver's static desired course, not to the position of the vehicle in the previous maneuver step!

#### Driver Parameter...

Clicking on this button opens the GUI in which the driver model can be parameterized. See the [section 4.7 'Driver' on page 116](#) to learn how to do it.

### Sinus

The *Sinus* option enables to apply a regular sinus input to the steering wheel. The usual properties of a sinus wave have to be parameterized.

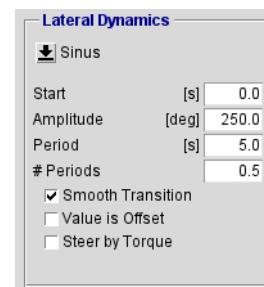
**Start (s)** It specifies when the sinus input should start. The value is calculated from the beginning of the maneuver step.

**Amplitude (deg) or (Nm)** The amplitude of the input is specified here. The amplitude is calculated from the null angle, not peak to peak.

The unit changes according to the steer model you chose: default steer by angle (unit: deg), or specific steer by torque (unit: Nm).

**Period (s)** The period of the sinus waves is specified here.

**# Periods (-)** The number of periods to be applied to the steering wheel is specified here.



**Smooth Transition** If this option is activated, the sinus wave ascension is smoothed:

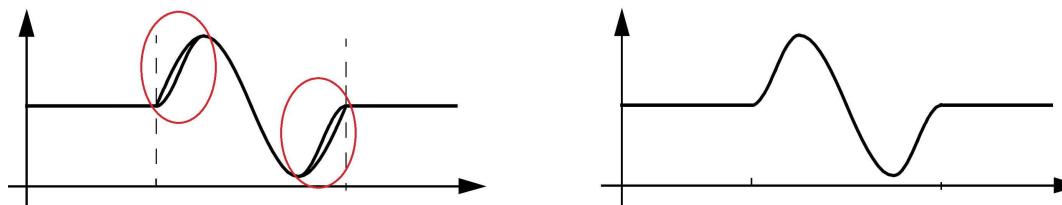


Figure 4.87: Sinus steering signal with *Smooth Transition* (left) and without (right)

**Value is Offset** If this option is activated, the value defined in "Amplitude" is an offset, calculated from the value of the steering wheel angle (respectively steer torque) at the moment when the sinus input starts to be applied.

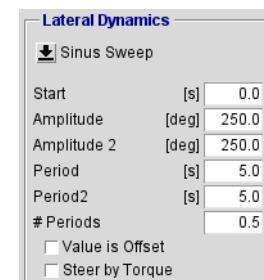
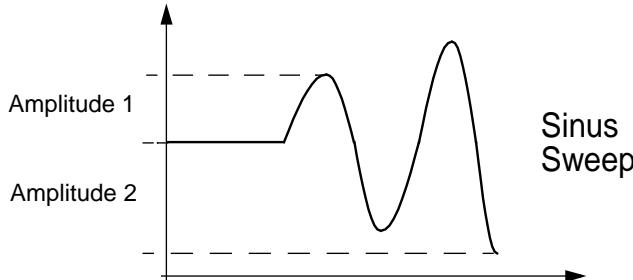
**Steer by Torque** If you use a model library with a steer by torque model (e.g. through a Simulink model), you have to activate this option. In all other cases, especially if you use the default model available in CarMaker, leave this option deactivated.

## Sinus Sweep

The *Sinus Sweep* option lets you insert an ascending sinus input to the steering wheel. Additionally to the ordinary sinus parameters described above in the section [Sinus](#) the following options are available here:

**Amplitude 2 (deg) or (Nm)** The final value of the ascending amplitude.

**Period 2 (s)** If you would like to sweep the period time of the sinus here the final period length is defined.



As an example take the TestRun Examples > VehicleDynamics > Sinus\_Dwell.



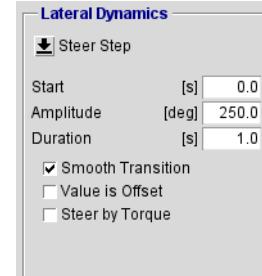
## Steer Step

The *Steer Step* option enables to apply a steady angle ramp (respectively torque) to the steering wheel, up to a defined angle (respectively torque) limit. This option can be used to apply a step input if the ramp is very short. The usual properties of a ramp have to be parameterized.

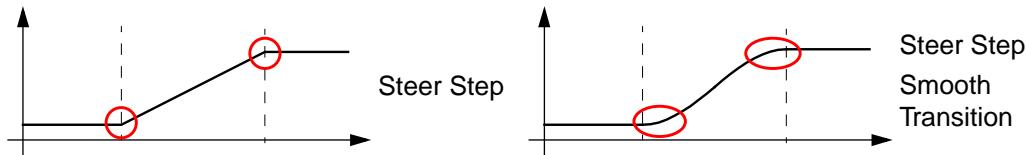
**Start (s)** It specifies when the ramp input should start (unit: s). The value is calculated from the beginning of the maneuver step.

**Amplitude (deg) or (Nm)** The amplitude of the input is specified here (unit: deg; if Steer by Torque, unit: Nm). The amplitude is calculated from the null angle, not peak to peak.

**Duration (s)** The duration between the beginning of the ramp (defined by *Start*) and the moment when the limit value is reached (defined in *Amplitude*) is specified here.



**Smooth Transition** If this option is activated, hard transitions between the steps can be avoided:

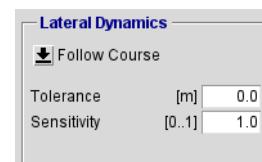


**Value is Offset** If this option is activated, the value defined under *Amplitude* is an offset, calculated from the value of the steering wheel angle (respectively steer torque) at the moment the input starts.

**Steer by Torque** If you use a model library with a steer by torque model (e.g. through a Simulink model), you have to activate this option. In all other cases, especially if you use the default model, leave this option deactivated.

## Follow Course

The *Follow Course* option activates a primitive steering wheel angle (respectively steer torque) controller. It is a controller which follows the center line of the road. It does not regard any course changes like pylons. It can be useful if you do not have too high accuracy requirements or if you would like to check the behavior of IPGDriver.



**Tolerance (-)** Sets the maximum deviation from the ideal line that is allowed.

**Sensitivity (-)** The controller exit is multiplied by this factor. It defines the gradient of the control parameter: it determines how fast the controller (in that case the driver) should react.

## 4.5.8 Minimaneuver Command Language

In the Maneuver dialog, you have access to the field *Minimaneuver Commands*. This field is specific for each maneuver step and enables to improve the maneuver definition thanks to a set of very easy commands.

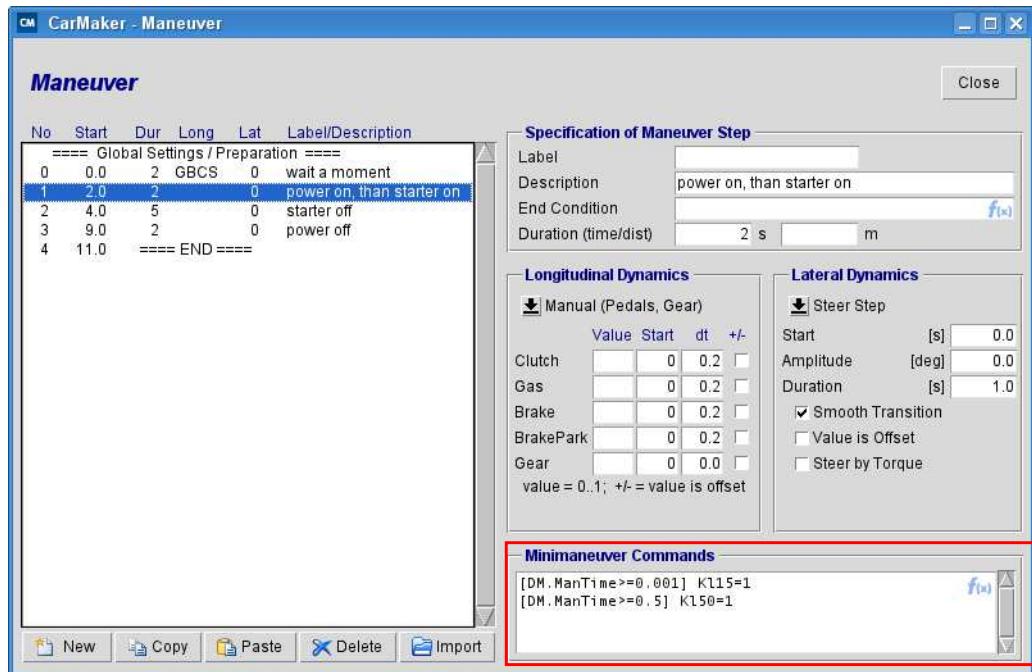


Figure 4.88: CarMaker Maneuver dialog with a minimaneuver command

All minimaneuver commands are written in the so called *Minimaneuver Command Language*, which compromises a small group of commands that can be specified in the Car-Maker Maneuver dialog and that will be executed during a minimaneuver.

There are commands that change quantities, parameter files, create new quantities, operate the FailSafeTester, write log entries, trigger the result storage and much more. Realtime Expressions (see [appendix 'Realtime Expressions' on page 602](#)) can also be used in the Minimaneuver Command Language.

### Syntax

The execution of Minimaneuver commands can be defined by a trigger event. The general syntax of a minimaneuver command looks as follows:

```
[Trigger] CmdStr
```

The trigger, indicated by squared brackets, is followed by a command string. Both are separated by a blank. The trigger is optional. If no trigger event is specified, the command string is executed at the start of the maneuver step. Realtime Expressions are indicated by the prefix *Eval*:

```
Eval RealtimeExpression
```

Comment lines start with a '#' character and are ignored by the interpreter. All other lines are interpreted as commands.

## Trigger

When the trigger conditions become true, the successive command string is executed. The trigger is always indicated by square brackets. All User Accessible Quantities, temporary variables and ScratchPad values can be combined with a logical expression which will be continuously evaluated during the simulation. Even more than one logical expression can be used at a time. The expressions need to be linked with a logical AND/OR. Apart from logical expressions all functions coming with, Realtime Expressions are available. The table below gives an overview of frequently used operators. Please find a complete list of all possible operators in the [appendix C.2 'Operators and Functions' on page 603](#).

Table 4.13 Operators for trigger conditions in minimaneuver commands

Logical operator	Condition is valid if
<code>&gt;=</code>	quantity is larger than value xy
<code>&lt;=</code>	quantity is smaller than value xy
<code>==</code>	quantity reaches the exact value xy
<code>&amp;&amp;</code>	logical expression 1 AND logical expression 2 are true
<code>  </code>	logical expression 1 OR logical expression 2 are true
<code>^^</code>	ONLY logical expression 1 OR ONLY logical expression 2 are true (XOR - exclusive logical OR)
<code>change(a)</code>	detect change of value a; true if value changed

**Example** Execute the minimaneuver command when the car reaches a velocity larger than 27 m/s:  
`[Car.v>=27] minimaneuver command`

**Example** Execute the minimaneuver command when the car reaches a velocity larger than 27 m/s AND a lateral acceleration of less than 2.5 m/s<sup>2</sup>:  
`[Car.v>=27 && Car.ay<=2.5] minimaneuver command`



Please note that the trigger conditions always use SI units. However, you can combine an User Accessible Quantity with a mathematical operator such as a multiplying factor to convert units:



**Example** `[Car.v>=100/3.6] minimaneuver command`

Using Realtime Expressions, the trigger condition can be set implicitly using an If-Then relation:

`Eval IfCondition=True ? ActionCommand`

**Example** If the simulation time has exceeded 50s press the brake pedal to 70%:  
`Eval Time>=50 ? DM.Brake=0.7`

For commonly used User Accessible Quantities the following aliases are available:

Table 4.14: Aliases for commonly used UAQs

Alias	Corresponding User Accessible Quantity
t	Time (absolute simulation time)
s	Car.Road.sRoad (absolute distance travelled from beginning of the simulation)
cnt	CycleCount (counter for simulation cycles)

**Example** For example, the following expressions are identical, the minimaneuver command is executed after a total distance of 50m is travelled:  
[s>=50] minimaneuver command  
[Car.Road.sRoad>=50] minimaneuver command

## Commands

Please find a detailed list of all possible Minimaneuver commands in the [appendix B.1 'Command Reference' on page 590](#).

## 4.6 Input From File / Using Real Measurements

CarMaker offers the possibility to use real measurements of gas pedal position, brake pedal position, steering wheel angle etc. to control the vehicle instead of defining longitudinal and lateral maneuvers that are executed by a driver model. This functionality is called Input from File (IFF) and can be activated via the CarMaker GUI > Parameters > Input From File.

In the Input from File GUI, a ASCII file containing measurements for steering wheel angle, steering torque, pedal position, gear number, park brake position and/or the pressure in the master brake cylinder is loaded. Each channel is linked to a suitable CarMaker quantity. Thus it overwrites the values of this quantity that are set in the maneuver definition. It is not mandatory to use measurements for all channels at the same time - the quantities that are not linked to a measurement keep the value set by the maneuver definition.



Please avoid setting the channel *Speed* at the same time as *Gas Pedal* and *Brake Pedal*. This may leads to contradicting inputs. *Speed* is designed to be used only with the longitudinal maneuver *Speed Profile* (see section 'Speed Profile'). In this case, IPGDriver follows the prescribed speed profile.

By clicking on the GUI using the right mouse button, a context menu opens. You can choose between *Reset-*, *Load-*, *Save Configuration*, *Import* and *Edit Aliases*. The configurations can be loaded from the project-, shared- and central directory folder *Sim\input\Settings*. The aliases remember frequently used names for each channel in the input file (e.g. *Vel* for the speed channel). They are used for automatic channel linking. To get an overview of the default aliases, have a look at [Table 4.15: Default Aliases \(case insensitive\)](#).

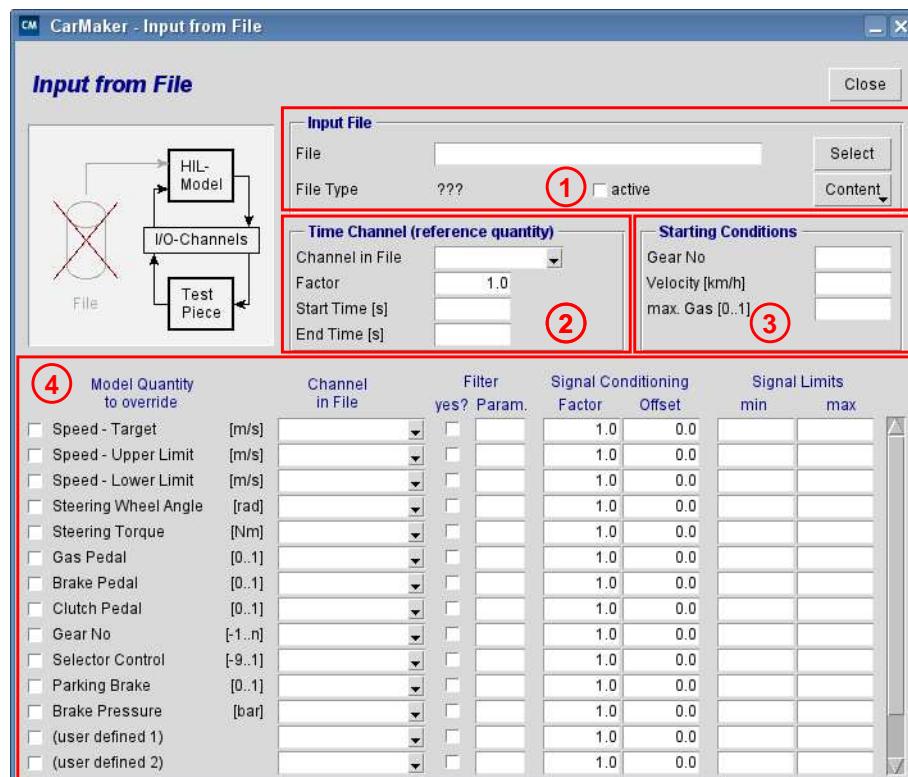


Figure 4.89: The Input from File dialog

## Activation of a Measurements File (Box 1)

- File** The file containing the measurement is loaded here. Only one file can be chosen, thus if the measurement is splitted into several files, you should gather them into a single one. This file must be a text file without commas or other punctuation marks and should consider the following syntax: Each column is referred to as a channel. Tabs are allowed to separate the columns of each channel:

The first line specifies the channel names. In an optional second line, the unit and/or a factor for each channel can be specified. The first character of both lines has to be a hash "#". #Name and #Unit are also possible. If both unit and factor are specified, they have to be separated with \* (Example: m/s\*1.5). If the unit and the factor are not known, there have to be set the placeholder - or 1.0. If no units and factors are known, the line has not to be specified at all.

```
# Time StWhl Speed Gas
# s - m/s*5 0.01
0.02 0.0 1.3 50
0.04 0.0 1.8 40
```

or:

```
#Name Time StWhl Speed Gas
#Unit s - m/s*5 0.01
0.02 0.0 1.3 50
0.04 0.0 1.8 40
```

This file has to be placed in the folder *SimInput* of your CarMaker project-, shared- or installation directory.

- File Type** This parameter is read only. It shows if the file is readable by CarMaker. If the format is readable, the name of the file is displayed nearby. If this is not the case, the comment "unknown" is displayed instead.

- Active** This option activates the loaded input file. If the measurement seems to be ignored in the simulation, check if this option is activated.



Note that IFF can also be activated during the simulation via the Minimaneuver Command Language ([section B.1.1 'Driving Maneuver Commands'](#)). In this case, keep in mind that there may be an offset between the IFF timeline and the maneuver timeline.

- Content** Clicking on this button shows the file which is currently loaded. By pressing this button for a while, two additional options will appear: *Edit File* (open file in text editor) and *Plot File* (plot the content of the file in IPGControl).

## Definition of the Input File Time Vector (Box 2)

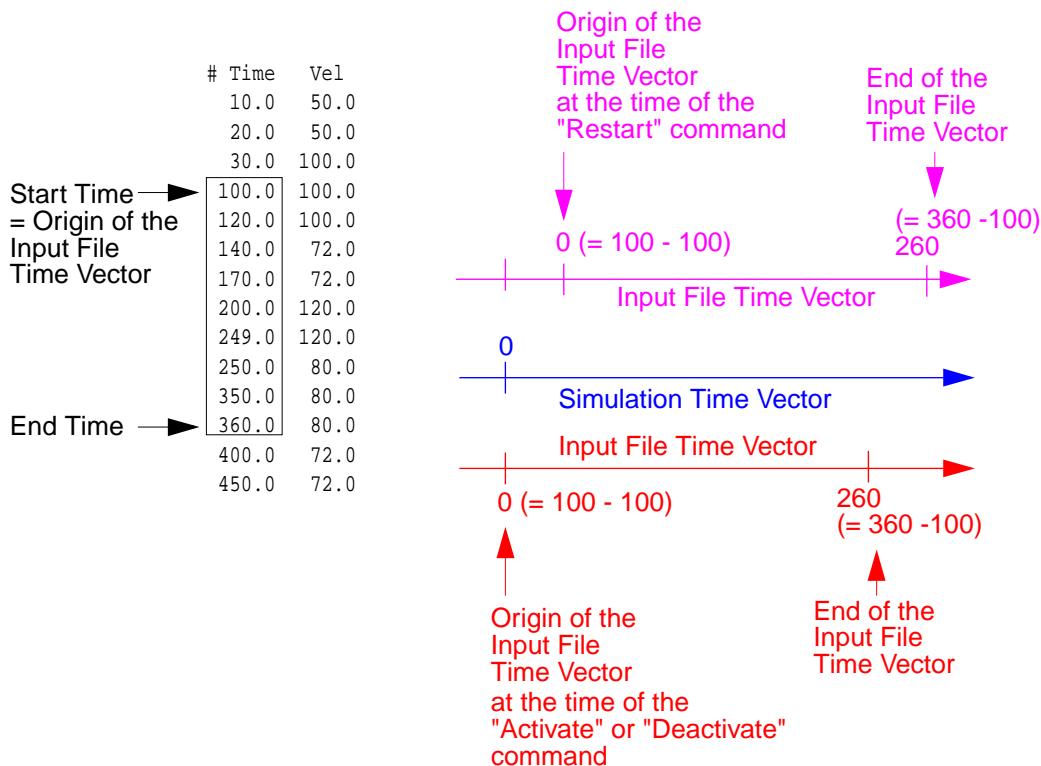
Time Channel (reference quantity)	
Channel in File	<input type="button" value="▼"/>
Factor	1.0
Start Time [s]	
End Time [s]	

Figure 4.90: Time Channel menu

- Channel in File** Any input file requires a time channel. Without this channel, the IFF functionality will not work. This is why the name of the time channel has to be identified in this case sensitive field. You can select available channels using a dropdown menu on the right side of "Channel in File". The time channel in the file can also start with another value than zero.

**Factor (-)** CarMaker requires a time channel in seconds. If this is currently not the case, you can define a multiplying factor manually in the GUI or the Input File (see syntax of input file). It is also possible to define the current unit in the input file and the factor for conversion in SI units (seconds) will be calculated automatically.

**Start Time** Optional. These fields are used, if only a part of the input file time vector should be used. For instance, let us suppose that the file contains a time vector ranging from 10s to 450s. If you would like to use the measurements in CarMaker only between 100s and 360s, enter 100 in the field *Start Time [s]* and 360 in the field *End Time [s]*.



The *Restart* function can be triggered only via the Minimaneuver Command Language (see section B.1.1 'Driving Maneuver Commands').

The *Activate / Deactivate* function can be triggered via Minimaneuver Command Language. Per default, IFF is automatically activated at the start of the maneuver.

#### Definition of the Starting Conditions (Box 3)

Starting Conditions	
Gear No	1
Velocity [km/h]	0.0
max. Gas [0..1]	0.5

Figure 4.91: Defining the starting conditions

As indicated by their name, the conditions Gear No, Velocity [km/h] and max. Gas [0..1] are used as initial values at the start of the input file

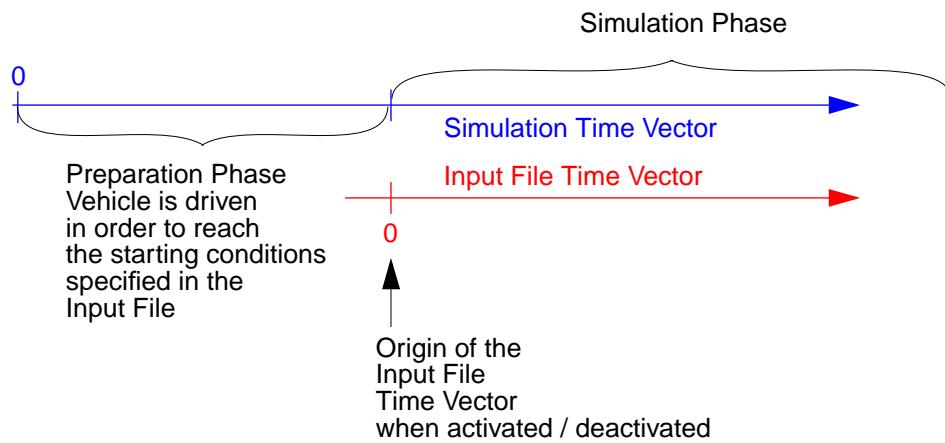


In order to have a steady situation at the start of the maneuver, CarMaker starts a procedure to reach these values. The parameterized maneuver does not start and the simulation status in the CarMaker GUI remains on "Preparation" as long as the Starting Conditions are not reached.

**Gear No** (-) This is the gear number at the beginning of the time vector of the input file.

**Velocity** (km/h) This is the vehicle speed at the beginning of the time vector of the input file.

**Max. Gas** (-) This is the gas pedal position (value between 0 and 1) at the beginning of the time vector of the input file.



#### Selecting and Formatting the Quantities to overwrite (Box 4)

Model Quantity to override	Channel in File	Filter yes? Param.	Signal Conditioning Factor	Signal Limits min	Signal Limits max
<input type="checkbox"/> Speed - Target	[m/s]	[ ]	0.4	0.0	
<input type="checkbox"/> Speed - Upper Limit	[m/s]	[ ]	1.0	0.0	
<input type="checkbox"/> Speed - Lower Limit	[m/s]	[ ]	1.0	0.0	
<input type="checkbox"/> Steering Wheel Angle	[rad]	[ ]	1.0	0.0	
<input type="checkbox"/> Steering Torque	[Nm]	[ ]	1.0	0.0	
<input type="checkbox"/> Gas Pedal	[0..1]	[ ]	1.0	0.0	
<input type="checkbox"/> Brake Pedal	[0..1]	[ ]	1.0	0.0	
<input type="checkbox"/> Clutch Pedal	[0..1]	[ ]	1.0	0.0	
<input type="checkbox"/> Gear No	[-1..n]	[ ]	1.0	0.0	
<input type="checkbox"/> Selector Control	[-9..1]	[ ]	1.0	0.0	
<input type="checkbox"/> Parking Brake	[0..1]	[ ]	1.0	0.0	
<input type="checkbox"/> Brake Pressure	[bar]	[ ]	1.0	0.0	
<input type="checkbox"/> BrakeForce	[N]	[ ]	1.0	0.0	
<input type="checkbox"/> (user defined 2)		[ ]	1.0	0.0	

Figure 4.92: The dialog of box 4 to edit quantities to be overwritten.

In the list of the quantities that may be overwritten, the *Speed* signal are treated differently. Whereas the other signals are overwritten directly, the *Speed* is the result of the vehicle model calculation. With Input From File, you can only define a target speed that IPGDriver tries to follow as close as possible. The channels *Speed - Upper/Lower Limit* can be used to define a speed range IPGDriver should maintain, e.g. like in a driving cycle. These channels are used by the longitudinal maneuver *Speed Profile* only.



The four last quantities are referred to as *user defined* channels. They can be used to set other quantities (e.g. activation of an driver assistance system) from measurements. This requires changes in the c-code interface of CarMaker.

#### Model Quantity to override

Clicking on the tick box in front of the quantity will activate the IFF functionality. For this quantity, the values set from maneuver definition are overwritten. The values for this quantity are read directly from the file and transmitted to the vehicle model.

#### Channel in File

For each parameter to be overwritten, the name of the corresponding channel in the file has to be specified. This field is case sensitive. You can select available channel names by using a dropdown menu at the right side of the *Channel in File* fields. It is possible to define channel aliases by clicking the right mouse button on the dialog and choosing *Edit Aliases*.

Table 4.15: Default Aliases (case insensitive)

Quantity	Aliases
Time	Time, t, Zeit
Velocity	Velocity, Car.v, Geschwindigkeit, Speed
StWhlAngle	DM.Steer.Ang, StWhlAngle, Lenkradwinkel, Steering-WheelAngle
StWhlTrq	StWhlTrq, Lenkmoment, SteeringTorque
Gas	Gas, DM.Gas, Gaspedal
Brake	Brake, DM.Brake, Bremspedal, BrakePedal
Clutch	Clutch, DM.Clutch, Kupplungspedal, ClutchPedal
GearNo	GearNo, DM.GearNo Gang
SelectorCtrl	SelectorCtrl, DM.SelectorCtrl, Wahlhebel, SelectorControl
BrakePark	Brake.Park, DM.BrakePark, Feststellbremse, Parking-Brake
pMC	pMC, Bremsdruck, BrakePressure
BrakeLever	Brakelever, Bremshebel, Brake.Lever
pMCPedal	pMCPedal, BrakePressurePedal, Bremsruckpedal, Brake.pMClever
pMClever	pMClever, BrakePressureLever, Bremsdruckhebel, Brake.pMClever

When opening a new input file, CarMaker checks the aliases list and proposes channels that fit the required quantities.

**Filter:** Input files that are directly derived from real world measurements maybe need to be filtered.  
**Yes?** This can be done by selecting the filter option (yes?) of IFF. Once this option is activated, the filtered value will be the average value of the current value and of the (n-1) values before. The value of n is set with the parameter *Param.*  
**(-)**

**Factor** CarMaker requires the quantity in SI units. If the input file data is provided in an other unit (e.g. deg instead of rad for the steering angle), *Factor* and *Offset* can be set for converting the data to SI units. For a wide range of units, the factor can be calculated automatically.  
**Offset**  
**(-)**

$$\text{Value} = (\text{Input} - \text{Offset}) * \text{Factor} \quad (\text{EQ 1})$$

#### Signal Limits Min / Max (-)

It is also possible to limit the values of a quantity read in a channel before being used in the simulation. The value which is limited is the value read in the channel after correction of the filter, multiplying factor and offset.



Please note that the signals for gas, brake and clutch pedal position are automatically limited to values between 0 and 1.



You can find examples in the folder Examples > CarMakerFunctions > ExternalInputs.

## 4.7 Driver

### 4.7.1 Overview of IPGDriver

You can reach the Driver dialog from the CarMaker Main GUI > Parameters > Driver:

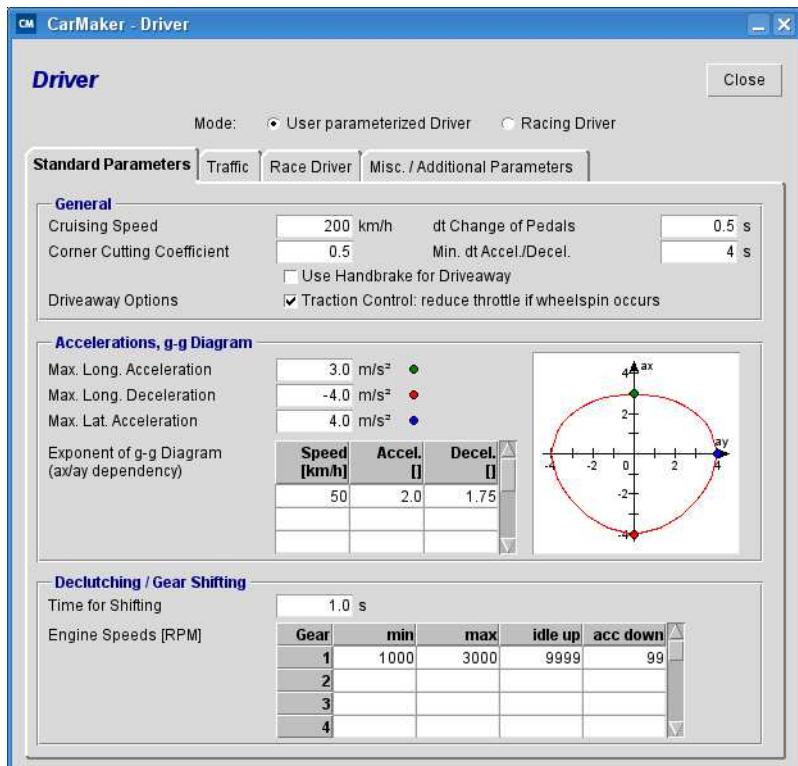


Figure 4.93: The Driver dialog box

Moreover, please refer to the IPGDriver User Manual as well.

#### Idea of IPGDriver

IPGDriver is a controller for following a course and a speed controller on a given track. It can be activated in the Maneuver dialog (see [section 4.5.6 'Longitudinal Dynamics' on page 100](#) and [section 4.5.7 'Lateral Dynamics' on page 103](#)).

IPGDriver enables you to add the control actions of a human driver to your complete vehicle simulation. These actions include the steering, braking, throttle position, gear shifting and clutch operation. But you can decide to use IPGDriver only to control the course and not the speed or vice versa (see the [section 4.5 'Maneuver' on page 91](#))! To sum up, the driver actions are:

- choice of driving course within the lane borders (corner cutting)
- steering
- choice of the driving speed according to the course and the vehicle behavior
- influence on the speed over the accelerator and brake pedal, as well as over the clutch pedal and gear selection
- regard traffic objects and follow



Note that the driving lane of the driver is defined via the road interface: please look at the [section 'Driving Lane' on page 42](#).

## Driver Model Versions

CarMaker includes two versions of IPGDriver:

- *User parameterized Driver*: This model is part of every version of CarMaker. The user parameterized driver can perform a driver adaption to automatically learn about the vehicle's limits by performing a pre-set sequence of maneuvers.
- *Racing Driver*: This model is available in the pro version of CarMaker only. In addition to the user parameterized driver, the Racing Driver enables to optimize the laptime in function of the speed and to perform a driver adaption procedure to automatically find these limits.

With the Racing Driver, note that special care must be taken to simulate a real driver with all its natural limits. That's why the model, for example, can acquire information about the vehicle on its own (learn) from the reactions of the vehicle. It does not get any information that can not be recognized by a real driver, even if that information is numerically available.

Before going further, you may already have a question: if I have CarMaker pro, which driver model should I choose?

The answer depends on the TestRun you want to simulate. If it requires to know at best the vehicle dynamics limits (lap time optimization), the Racing Driver is recommended. But if you just want to accelerate up to a steady speed before starting an open loop maneuver (braking), or performing some special maneuvers (accelerating very slowly on a steady circle, ...), then the User parameterized Driver model is enough, and in fact faster and easier to parameterize.

The usage of the right mouse button is very helpful. A small window like in [Figure 4.94](#) offers options to import a complete driver definition as well as the knowledge from a different TestRun. Because sometimes it is not clear how a real driver can be characterized, you can load three different predefined driver setup proposals, a defensive driver, a normal driver and an aggressive driver. In the lower section you can make a reset of all driver parameters and you can clear all knowledge. After this, the default values of IPGDriver are used.



Figure 4.94: IPGDriver GUI with Right Mouse Button Click

## Full Documentation of IPGDriver

Complete information about the driver model can be found in the IPGDriver Manual: in the CarMaker GUI, click on > Help > Other IPG Products > IPGDriver. You will find in particular:

- a more detailed overview of the driver model (see the [section 'Getting Started'](#)),
- the calculation of the static desired course and the desired speed (see [section 3.1 'Course Selection'](#), [section 3.3 'Steering'](#), [section 3.2 'Speed Selection'](#) and [section 3.4 'Influence on Speed'](#)),
- the explanation of all parameters (see the [section 'Using IPGDriver'](#)),
- all details about the driver adaption (see the [section 3.5 'Basic Learning Procedure \(Manual Transmission\)'](#), and [section 3.6 'Basic Learning Procedure \(Automatic Transmission\)'](#)).



You can find examples in the folder Examples > CarMakerFunctions > IPGDriver.

## 4.7.2 User parameterized Driver

First of all you have to choose the right Driver Mode (top of the Window).

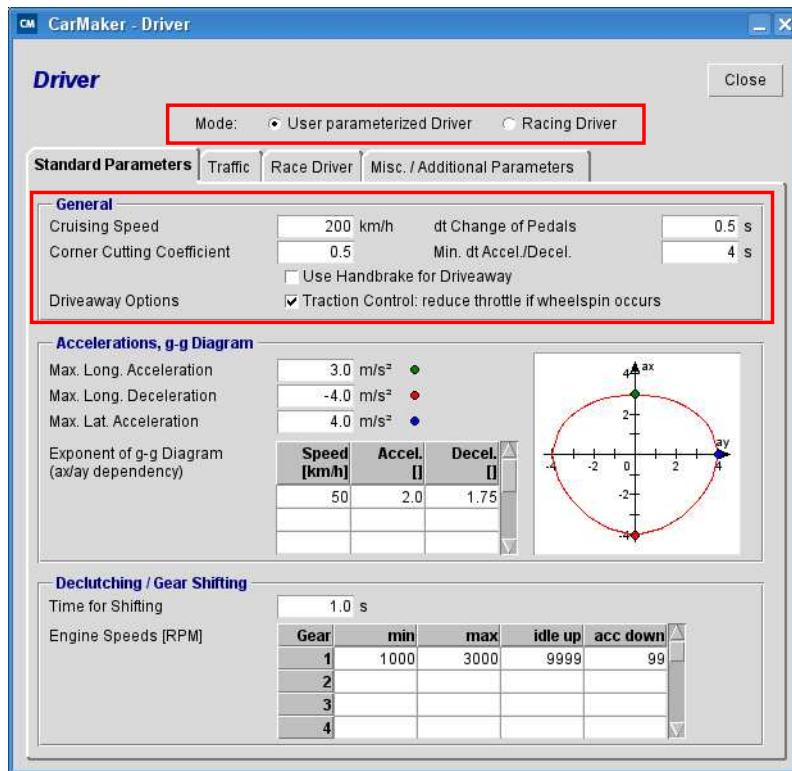


Figure 4.95: Parameters of the Driver GUI

**Mode** Activate the option "User parameterized Driver" to switch to this driver model. Once this mode is activated, you can parameterize the driver model in the tab "Standard Parameters".

Additional parameters are also available in the tab "Additional Parameters", but at first they are of lower importance.

### Cruising Speed (km/h)

The specified speed is the target that IPGDriver tries to reach.

He will of course control the vehicle, brake it if necessary in order to keep it on the road according to the limits he learned about this vehicle. But it will always try to accelerate to reach this speed.

### General

#### Corner Cutting Coefficient (-)

This is one of the parameters you already tested by reading the instructions of the Quick Start Guide. This parameter allows the driver to move off the given driving lane. The values of the Corner Cutting Coefficient range from 0 to 1. Zero means the vehicle drives exactly in the middle of the driving lane, and 1 means the driver uses the whole width of the driving lane calculate its static desired course.

For instance, if it is told to drive on the center of the road (see [section 'Driving Lane' on page 42](#)), and if you define a corner cutting coefficient of 0, it will follow exactly the middle line of the road.

#### dt Change of Pedals (s)

It is the time required by the driver to move its foot from the throttle to the brake pedal. If IPGDriver switches from the throttle to the brake it waits this duration (starting after the throttle value equals zero) before allowing the brake value to change. This also applies to the other direction (e.g. brake to throttle).

- Min. dt Accel./Decel. (s)** It is the time the driver is cutting speed peaks in the static desired speed because of too short acceleration and deceleration sequences. Shortening this value will usually improve a lap time.

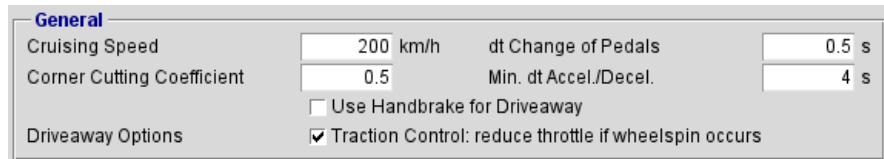


Figure 4.96: Influencing the driver's characteristic

- Use Handbrake for Driveway** With this option the driver will release the handbrake when starting off.

- Traction Control** IPGDriver includes an internal traction controller. When this tick box is activated, IPGDriver automatically reduces the throttle when wheel spin occurs. Per default it is activated with using the internal traction control function during the starting procedure. If deactivated, IPGDriver will push the gas pedal to gain speed, also with spinning wheels. It is useful for investigations of the ESP-TC function of the car.

## Accelerations, g-g Diagram

**Max. Long. Acceleration**  
**Max. Long. Deceleration**  
**Max. Lat. Acceleration (m/s<sup>2</sup>)**

These three values specify the maximum absolute acceleration (purely longitudinal or lateral).

Only one value is necessary for the lateral acceleration (value taken for a left curve), but two are required for the longitudinal acceleration and deceleration. The value for the lateral acceleration needs to be positive, but the value for the longitudinal acceleration can be either positive or negative.

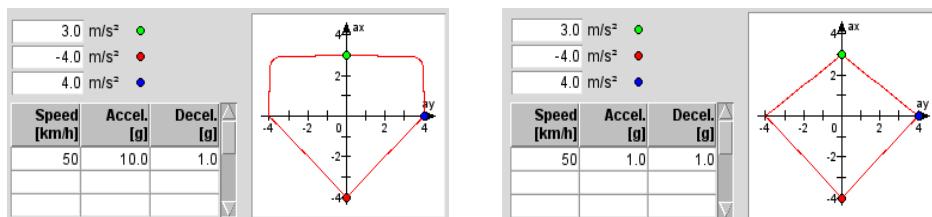
Those maximum values are represented on the g-g diagram by the green, red and blue points on the axis.

**Exponent of g-g Diagram (-)**

These exponents define the dependency between the lateral and the longitudinal acceleration (column *Accel*) and between the lateral and the longitudinal deceleration (column *Decel*). The exponents can be observed on the *g-g diagram* thanks to the red curves that link the maximum longitudinal and lateral accelerations.

The driver will drive so that the accelerations applied to the vehicle remain in the area defined by the exponents and the maximum limits (see next picture).

A high value (10) will increase the area, and a low value (1) will reduce it.



Exponent for the acceleration is set to 10.  
 Exponent for the deceleration is set to 1.

Exponent for the acceleration is set to 1.  
 Exponent for the deceleration is set to 1.

Contrary to the maximum accelerations (User parameterized Driver only), the exponents can be specified for various speed levels (Column *Speed [km/h]*). If you do so, between two speed levels, the exponents used by the driver are those of the nearest speed level. E.g., if you defined two different exponents for 80km/h and 150km/h, the driver will consider the exponents defined for 80km/h at a speed of 100km/h.

Please note, that the driver's target speed and acceleration characteristics can be manipulated during the simulation via DVA. Further information can be found in the IPGDriver Reference Manual, section Use Cases: Online Modification of the SDV.

## Declutching /Gear Shifting

Declutching / Gear Shifting					
Time for Shifting					
Engine Speeds [RPM]					
	1.0	s			
Gear	min	max	idle up	acc down	
1	1000	3000	9999	99	
2					
3					
4					

Figure 4.97: Influencing the driver's shifting and clutching behavior

Here, the shifting points used by IPGDriver are defined. This table is only valid in case of a vehicle with manual gearbox or with an automatic gearbox that allows tiptronic (see [section 5.20.3 'Parametrizing the TCU' on page 268](#)).

At the same time, the shifting needs to be activated in the Maneuver dialog, as explained in [section 4.5.6 'Longitudinal Dynamics' on page 100](#).

You do not need to activate this option if you are using an automated gearbox. To simulate the driver's shift intervention at automated gear boxes (Tiptronic), though, you need to use the DVA mechanism (see [section 6.3 'DVA: Online Manipulation of the Simulation' on page 357](#)). This tab will be of no effort.

**Time for Shifting (s)** The time required by the driver to shift the gears is specified here.

**Engine Speeds (rpm)** For each gear, the optimum engine speed for shifting can be parameterized. If these values are the same for all gears, then you need to parameterize them only for the first gear, and leave the fields for the other gears empty.

- *min*

It is the speed at which the driver will shift down (e.g. while slowing down).

- *max*

It is the speed at which the driver will shift up (e.g. while accelerating).

- *idle up*

This parameter is used to improve the driver reaction on a race track.

It defines the range of engine speed that has to be maintained during constant speed.

It is first determined whether it is possible to drive at that constant speed in the near future. If this is possible, IPGDriver will then check if the engine speed will fall below the upshifting engine speed when shifting into the next higher gear. If the engine speed will not fall, it shifts into the next higher gear; otherwise it will keep the gear that is currently engaged.

- *acc down*

This parameter is used to improve the driver reaction on race track.

It serves to manipulate the acceleration strategy. If during the forecast in the shifting module IPGDriver finds it will be accelerating soon, the possibility of downshifting is checked. If the engine speed will be smaller than the speed given in *acc down*, it will engage the next lower gear.



As an example take the TestRun Examples > CarMakerFunctions > IPGDriver > Driver\_UserParameterized.

### 4.7.3 Traffic

IPGDriver is able to recognize traffic objects driving ahead. If the tick box *Consider Traffic* is activated, he will follow the traffic objects in the defined manner:



Figure 4.98: Definition of follow traffic behavior of IPGDriver

The control strategy can be influenced with several parameters, e.g. to follow exactly within a constant time gap. Furthermore it is possible to give the driver a rather economic, fuel saving characteristic during column driving or to allow him a dynamic behavior which leads to increased fuel consumption.

**Min./Max. Time Gap (s)** The Time Gap variables are speed dependent distances for realistic driving. The driver will try to keep the time gap to the leading vehicle within these limits.

**Min. Distance (m)** This parameter defines the minimum distance to a parking traffic object. Once the road gap is smaller than this value, IPGDriver will bring the vehicle to a standstill.

**Max. Distance (m)** With this value, the absolute limitation for the follow traffic function is specified. Once the road gap to the leading vehicle is larger than this value, IPGDriver switches over to the normal driving mode.

**Energy efficient driving** By this option the fuel consumption can be influenced remarkably. The values of the Energy Efficient coefficient range from 0 to 1. Zero stands for a very hurried, agile driver who permanently changes its velocity and gap to the leading vehicle. The economic driver with a coefficient set to 1 rather adopts to the driving behavior of the leading car. Thus, the average velocity is lower than with the nervous driver, but there are also fewer stopping phases. All in all the difference in fuel consumption between the different driver characteristics can make about 30%. Please find more information in the IPGDriver Manual.

**Overtaking** Flag that enables overtaking of slower traffic objects.

**Overtaking Ratio (0..1)** The overtaking ratio is a factor to scale aggressivity of overtaking. Set to 1, the driver will overtake in any situation, even if the maneuver cannot be foreseen and is very dangerous. At a ratio of 0, the driver will act carefully and he will overtake only in safe situations.

**Min. Speed Difference (km/h)** This is the minimum speed difference to the leading vehicle the driver tries to reach before he starts overtaking. This parameters overrules the driver's cruising speed and possible speed limitation defined in the road configuration.

As an example take the TestRun Examples > CarMakerFunctions > IPGDriver > Driver\_TrafficStopAndGo or Examples/CarMakerFunctions/IPGDriver/Traffic\_Overtake.



#### 4.7.4 Racing Driver

With this mode, only a few values have to be parameterized (see hereunder). Indeed, most of the parameters available with the simple driver model are calculated by CarMaker during the so called driver adaption, in particular the maximum longitudinal and lateral acceleration. Once the race model is activated, the values that do not need to be parameterized anymore are not available. The way you can get those values is explained in [section 4.7.5 'Learning Procedure: Driver Adaption'](#).

Additional parameters are also available in the tab *Additional Parameters*, but at first, they are of lower importance.

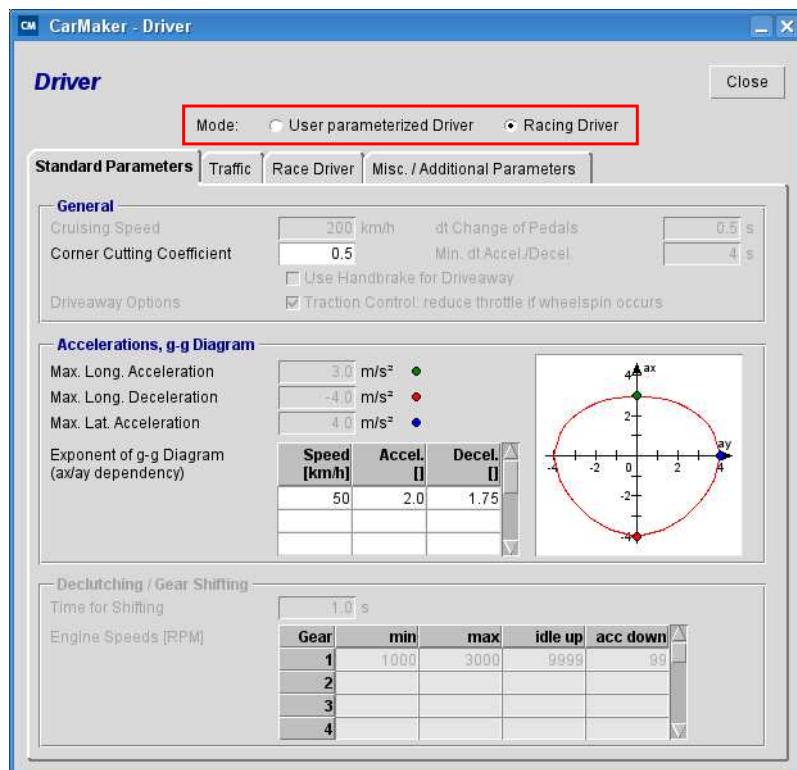


Figure 4.99: Standard parameters available for the Racing Driver

#### Parameters

To switch to the Racing Driver the corresponding mode has to be activated (top of the window).

**Mode** Activate the option *Racing Driver* to switch to this driver model.

**Corner Cutting  
Coefficient  
(-)**

This parameter remains available to match the driver's behavior to your needs. Its definition is the same as with the *User parameterized Driver*.

**Exponent of g-g  
Diagram  
(-)**



This parameter remains available to match the driver's behavior to your needs. Its definition is the same as with the User parameterized Driver.

As an example take the TestRun Examples > CarMakerFunctions > IPGDriver > Driver\_RaceMode.

## 4.7.5 Learning Procedure: Driver Adaption

In order to calculate the vehicle's limits for the *Race Driver*, you have to start a so called *Driver Adaption*. In the CarMaker GUI, click on Simulation > Driver Adaption.



Figure 4.100: Dialog box of the Driver Adaption

Two different adaptions are proposed:

**Basic Knowledge**

This mode should always be selected. As you see you can activate two modules. Usually, we recommend you to activate both of them:

- Vehicle Limits

This module calculates the maximum longitudinal acceleration, deceleration and lateral acceleration of the vehicle.

- Controller Dynamics

This module calculates the time preview used by the driver to predict his actions during a TestRun.

- Engine speeds for shifting

This module calculates the optimal engine speeds for shifting.

**Race Driver  
Adaption**

This is a mode for some special competition use. With this, you can optimize a lap time on a given track. However, it is not recommended to use this adaption since it is worth only in a few special cases.

Once the adaption was finished, the calculated values are displayed in the tab *Race Driver* of the Driver GUI. If you think that some values are not correct, you can correct them here.

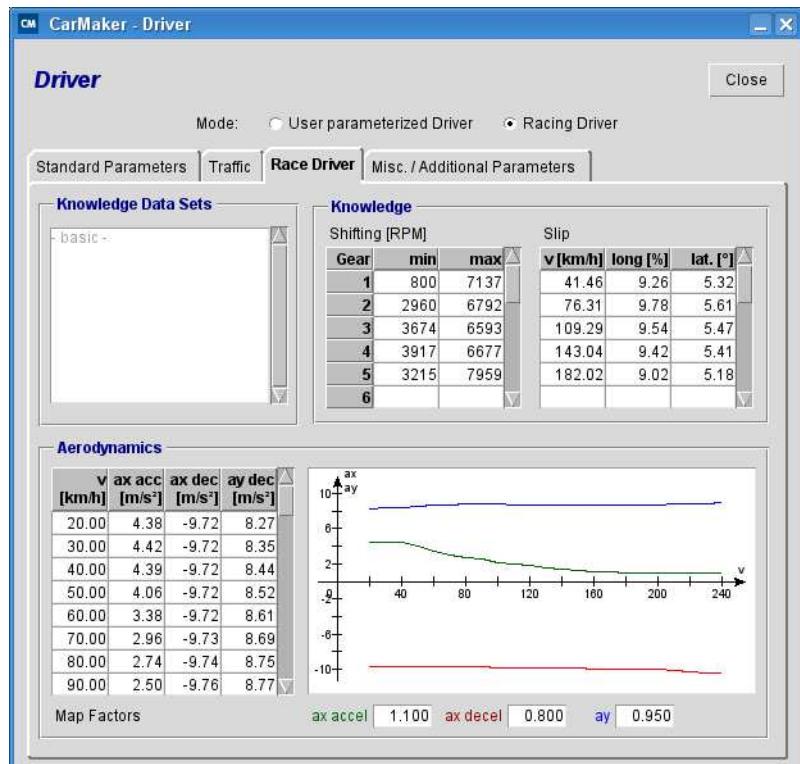


Figure 4.101: Racing Driver parameters



Do not forget to save the TestRun once you have completed a driver adaption, otherwise you will have to perform it again the next time you want to simulate!

This means that you can not perform more than one driver adaption per TestRun.

Basically, it needs to be done when the vehicle behavior has changed. Of course, a minor change of the vehicle which has a limited impact on its dynamical behavior does not require a new adaption, except if you want a great accuracy (but this implies that the entire vehicle is parameterized with an absolute accuracy and that this parameterization is validated). On the other hand, if you perform a change that has a big impact, such as other tires, or a completely new suspension, then a new adaption is clearly obliged. Otherwise, IPGDriver will try to drive with a given knowledge which does not match any longer to the vehicle's true behavior.

If you need more information about the driver adaption, you should read the IPGDriver Manual, section 3.5 'Basic Learning Procedure (Manual Transmission)' or section 3.6 'Basic Learning Procedure (Automatic Transmission)'. You can optionally contact your sales partner at IPG and let you know about the learning possibilities for IPGDriver.

## 4.7.6 Additional Parameters

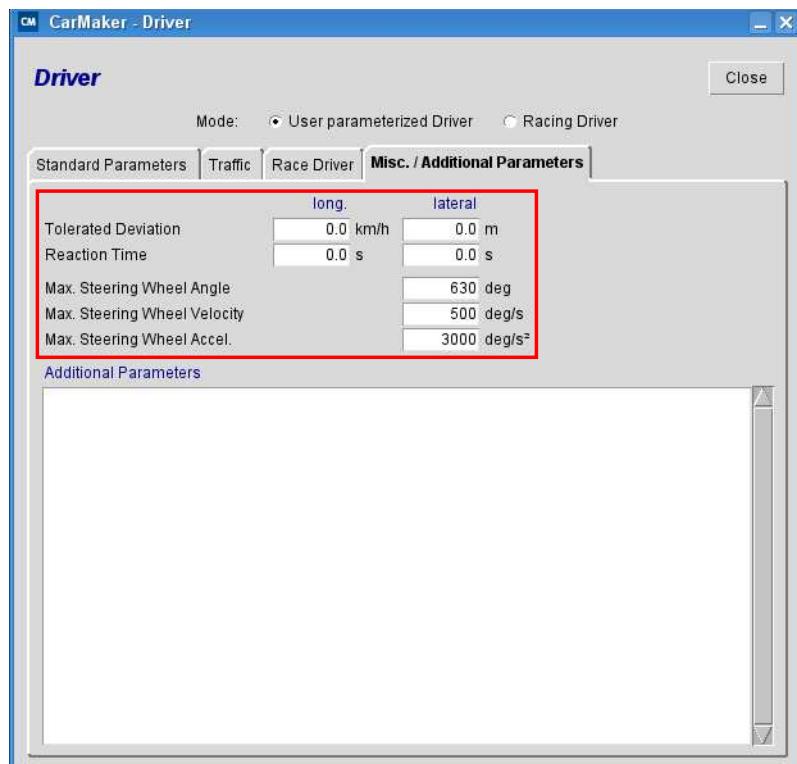


Figure 4.102: The third tab of the Driver GUI

In the GUI, the most important parameters of the driver model are available. In addition to these ones, IPGDriver can be tuned with other parameters which are available under the tab *Misc. / Additional Parameters*.

Those parameters are really for those who need to tune the driver's behavior, and correspond to a Power User level! A usual CarMaker user should not need to use these parameters.

- Tolerated Deviation long. (km/h)** This parameter is available for the *User parameterized Driver* only. It describes the threshold that IPGDriver uses to make corrections with the gas and brake pedals. If the absolute value of the predicted speed deviation is lower than this threshold, IPGDriver's activities are kept to a minimum, since corrections that would be made if the threshold was reached would be avoided. For studies requiring the given speed profile to be kept as exact as possible the threshold value should be set to 0.0.
- Tolerated Deviation lateral (m)** This parameter is available for the *User parameterized Driver* only. It describes the threshold for IPGDriver's activities. If the absolute value of the predicted deviation is lower than this threshold, the work that needs to be performed by IPGDriver is kept to a minimum. For studies that demand that a given course is followed as exact as possible, the threshold (like the corner cutting coefficient) should be given a value of 0.0.
- Reaction Time long. (s)** This parameter is available for the *User parameterized Driver* only. It enables you to influence the behavior of IPGDriver so that after exceeding the tolerated speed deviation IPGDriver waits for the reaction time before acting. This makes it possible to simulate the startle behavior of a driver. For the reaction time to respond, the value of the tolerated speed deviation has to be larger than 0.54 km/h.

<b>Reaction Time lateral (s)</b>	This parameter is available for the <i>User parameterized Driver</i> only. It allows you to influence the behavior of IPGDriver so that after exceeding the tolerated transversal deviation, IPGDriver waits for the specified reaction time to pass by before acting. This makes it possible to simulate the startle-behavior of a driver. Note that for the reaction time to respond, the value of the tolerated transversal deviation has to be larger than 0.15 m.
<b>Max. Steering Wheel Angle (deg)</b>	This parameter specifies at which point the driver will stop to turn the steering wheel, even if the vehicle is still not in the desired direction.
<b>Max. Steering Wheel Velocity (deg/s)</b>	It defines the maximum steering wheel velocity that can be achieved.
<b>Max. Steering Wheel Accel. (deg/s<sup>2</sup>)</b>	It defines the maximum steering wheel acceleration reachable.
<b>Additional Parameters</b>	These parameters can be written directly in the field of the window <i>Additional Parameters</i> .

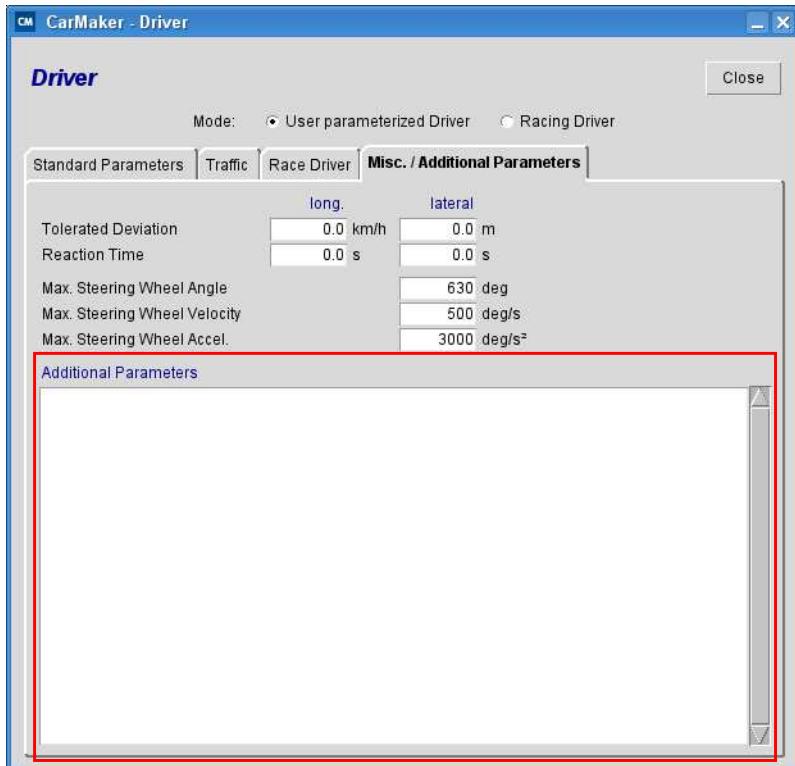


Figure 4.103: Entering additional parameters to the Driver GUI

A list of all additional parameters can be found in the Reference Manual of IPGDriver.

These additional parameters are especially used to tune the driver's course or to have technical information about the driver at the start off and during a simulation.

## 4.8 Traffic

### 4.8.1 Overview of IPGTraffic

IPGTraffic enables to simulate common traffic situations. For this, other road users can be added to a TestRun. These road users can be of any kind:

- parking vehicles,
- slow driving cars,
- vehicles that change the driving lane abruptly,
- cars that coincide at a crossing,
- bicycle riders,
- pedestrians crossing the road etc.



Figure 4.104: Every day traffic situations can be modeled using IPGTraffic

All these road users further referred to as *traffic objects* can have the following characteristics:

- A three-dimensional body which can have a geometry assigned by e.g. an object file.
- Moving or standing.
- Performing driving maneuvers independently from other road users.
- Existing for a given period of time.
- Randomly appear and disappear.
- Motion attributes:
  - maximal velocity,
  - maximal longitudinal acceleration and deceleration,
  - maximal lateral acceleration.

## 4.8.2 Defining Traffic Objects

To insert a traffic object into your TestRun, you need to open the Traffic Objects dialog (Car-Maker GUI > Parameters > Traffic Objects):

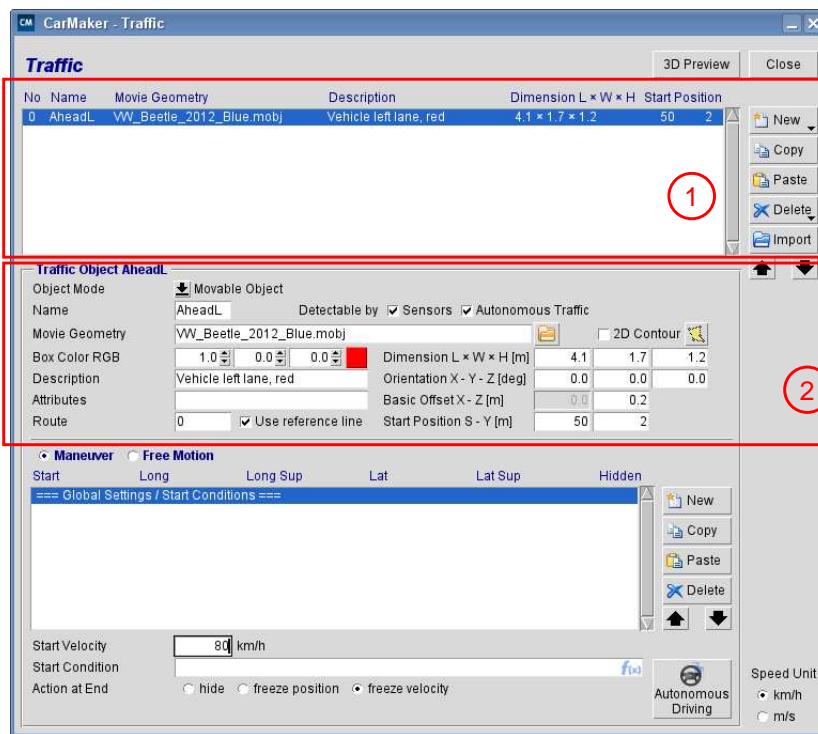


Figure 4.105: Traffic objects dialog

Box 1 in [Figure 4.105](#) lists all traffic obstacles of the current TestRun. The buttons at the right side can be used to add a new traffic object as well as to copy, paste or delete an existing one. The most important information about the traffic object is summed up in the columns:

- object number
- object name
- the movie geometry file used to easily identify the object
- a short description
- the basic dimensions
- the start position in road coordinates (s = longitudinal direction along center line; y = lateral deviation from center line).

In the middle section (box 2), the single attributes of each traffic object are specified. The following options are available:

## Attributes of Traffic Objects

- Object Mode** This dropdown menu enables you to select three different object modes. You can select between a *movable object* with predefined motion and two kinds of stationary objects. The mode only effects number of output quantities generated:

Table 4.16: Available modes for Traffic Objects

Object Mode	Description
Movable Object	Movement defineable either Maneuver based or with DVA, full set of User Accessible Quantities generated.
Stationary Object (with name)	Object will not move, only start position can be defined. Non-motion based User Accessible Quantities generated.
Stationary Object (without name)	Object will not move, only start position can be defined. No User Accessible Quantities generated.

If you are going to define a lot of stationary objects, you should go with the *stationary objects without name*. For these objects, no quantities are created which would limit the amount of objects at a certain point.

- Name** In this field you can give the object a name which will be used to refer to this traffic object. All User Accessible Quantities related to a single traffic object include its name string: `Traffic.Name.Parameter` (e.g. `Traffic.AheadL.sRoad`).

- Detectable by** Choose if the traffic object should be detectable for Driver Assistance Sensors and Free Space Sensors of the test car (option: "Sensors", see [section 5.23 'Sensors' on page 282](#)) and/or for other traffic objects with autonomous driving (option: "Traffic", see [section 4.8.5 'Autonomous Driving' on page 139](#)) autonomous traffic).

- Movie.Geometry** With this option, a three-dimensional movie object is stripped over the traffic object for animation purposes. In the same way as for the vehicle, a traffic object can be displayed in IPG-Movie by diverse geometries. The following two file formats are supported: \*.obj and \*.3ds. Example graphics can be found in the CarMaker installation directory under `..../IPG/hil/<version>/Movie`.



This option enables to insert any kind of object file. For example, you can use it to beautify the environment by inserting static traffic objects (velocity = 0) which are located besides the road. By loading an object file which contains the visual information of a building or a traffic sign you can easily build up a whole city. As an example, take the TestRun Examples > DriverAssistanceSystems > Autonomous Functions > PedestrianDetection\_CrossTheRoad. The whole city is built out of traffic objects.



Figure 4.106: City built with static traffic objects

#### Box Color RGB

Whereas the Movie Geometry is just an animation feature, the traffic objects are truly simulated as boxes with varying color and dimension. Just change the view mode to *ABRAXAS* in IPGMovie (see [section 'ABRAXAS' on page 402](#)) and compare.

To better differentiate between the single boxes, each can be given a different color, defined in RGB. In case you do not specify a Movie Geometry file the colored box is displayed instead.

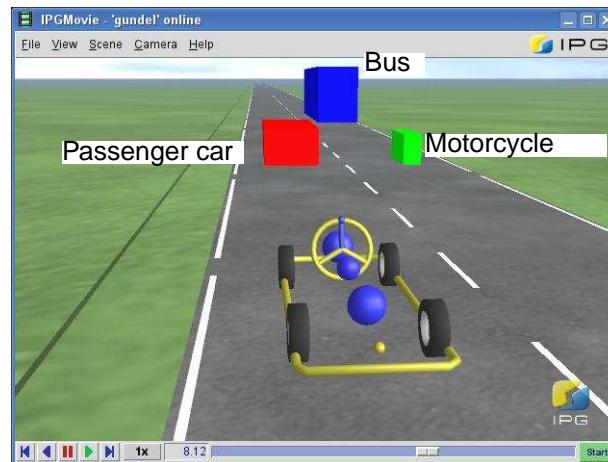


Figure 4.107: Traffic objects of varying color and size in the ABRAXAS mode of IPGMovie



Hint: By clicking in the color preview box next to the RGB definition fields you can select pre-defined colors.

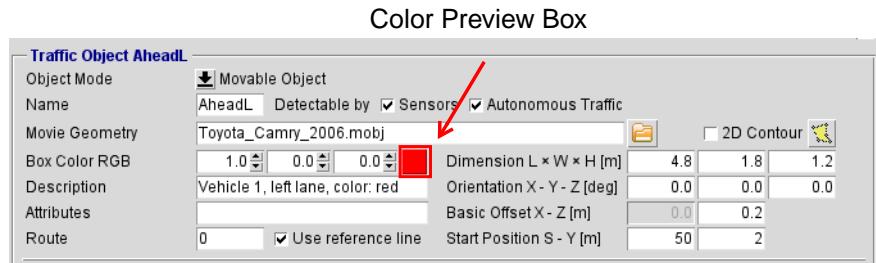


Figure 4.108: Defining the box color

- Description** At this point, a description with additional information about the traffic object can be entered.
- Attributes** With this option you can assign up to 10 attributes to a traffic object which are evaluated by your sensor model for instance. Thus, these additional attributes can be used to deposit reflection or deflection characteristics of the traffic object. All attributes need to be declared in the c-code interface. The single attribute items are separated by a blank.
- Route** In case a road network is available, each traffic object can follow an individual route. The routes have to be defined in the Road dialog (see [section 4.4.5 'IPGRoad 5.0: Road Networks'](#)).  
For oncoming traffic simply select the route that corresponds to the reverse route of the ego vehicle.
- Use reference line** In case, the route specifications do not include a path or the path is not relevant for the object, it can stick to the center line of the current lane.
- Dimension L x W x H (m)** These parameters define the overall dimensions of a traffic object as simulated in CarMaker, in the following order: length, width, height. All sizes are given in [m]. You can display the boxes by changing the view mode to *ABRAXAS* in IPGMovie (see [section 'ABRAXAS' on page 402](#) and [Figure 4.108](#)).

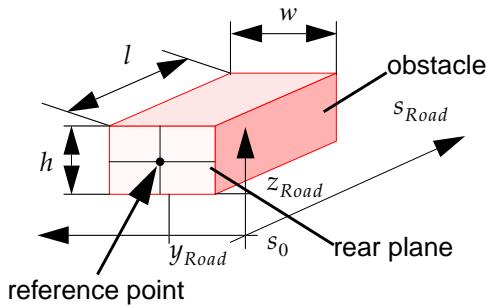


Figure 4.109: Overall dimensions of a traffic object in 3D view



Predefined traffic object classes with different basic shapes and pre-defined 3d graphics are available. When you keep the left mouse button pressed on the *New* button at the right side of the dialog, a selection menu with the predefined traffic object classes pops up:

Table 4.17: Basic shapes of traffic objects

Traffic Object Class	Length [m]	Width [m]	Height [m]	zOffset [m]
Compact Car	4.1	1.7	1.2	0.2
Sports Car	4.8	1.8	1.2	0.2
Rigid truck	8.5	2.5	3.2	0.5

Table 4.17: Basic shapes of traffic objects

Traffic Object Class	Length [m]	Width [m]	Height [m]	zOffset [m]
Semi Truck	18.0	2.5	3.2	0.5
Bus	10.5	2.5	3.0	0.3
Coach	12.0	2.5	3.8	0.3
Motorcycle	2.0	0.6	1.0	0.2
Bicycle Rider	1.8	0.7	1.7	0.0
Pedestrian	0.5	0.6	1.85	0.0

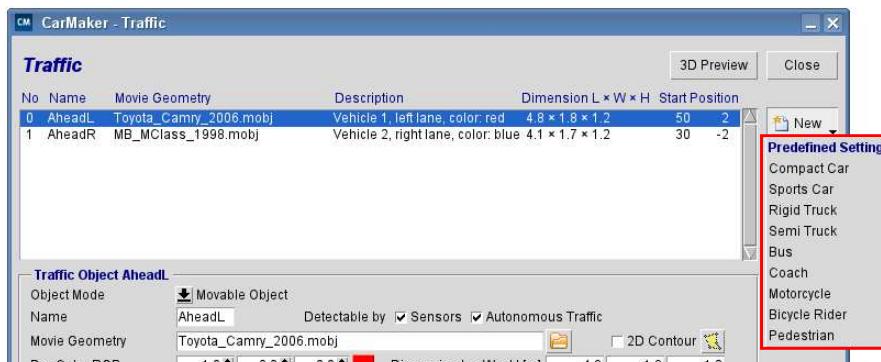


Figure 4.110: Selecting predefined traffic object classes

**2D Contour** Activating this tick box, a contour of the traffic objects can be displayed instead of the rectangular box. You do not need to define the length and width of the traffic object anymore in the main traffic GUI, just the height of the object is required.

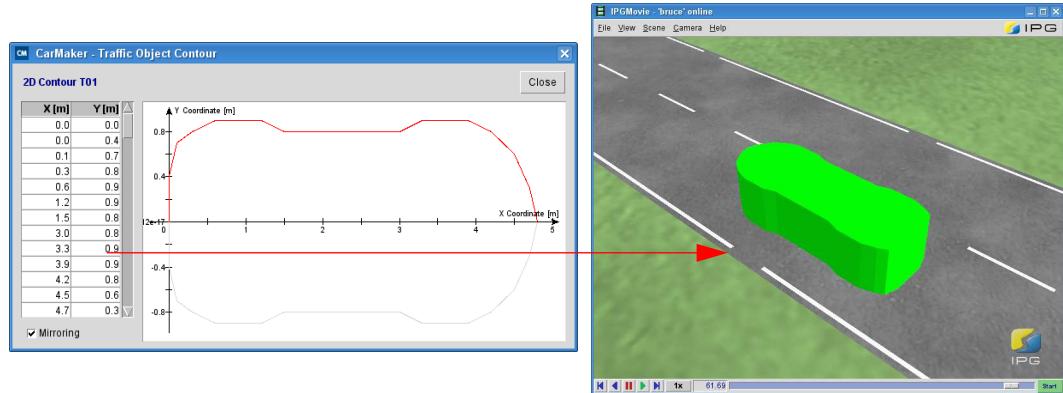


Figure 4.111: Defining the contour of a traffic object

To define the dimensions of the contour, click on the button right beside 2D Contour where you will find a table of x- and y-values. It is possible either to define an asymmetric contour or to mirror one side of the traffic object.

**Orientation X - Y - Z (deg)** At this point you can specify the initial orientation of the traffic object on the road. The angles around the x-, y-, z-axis are given in [deg]. The rotation sequence is z-y-x.

Rotations around x- and y- direction are restricted to small angles only in order to map vehicles parking on pavements for example.

**Example**      Orientation = 0 0 0



Figure 4.112: Simulating traffic

**Basic Offset Z (m)** Additionally, a basic offset can be applied to the traffic object. This parameter is mainly important for the sensor detection. At big vehicles like trucks, the clearance from the ground often is so high that the sensor can look underneath it. For such an application, a z-offset can be used which defines the distance of the traffic object from the road surface in [m]. Zero means the traffic object lies on the road surface, positive values move it upwards.

**Start Position S Y (m)** Start position of the traffic object in road coordinates ( $s$  = longitudinal direction along center line;  $y$  = lateral deviation from road center line). The coordinates refer to the object's rear plane.

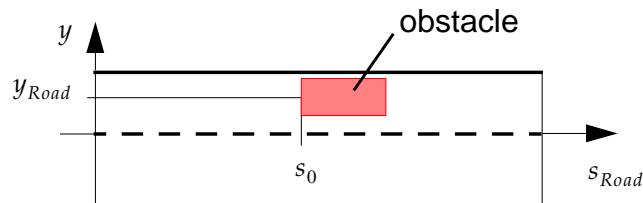


Figure 4.113: Positioning of a traffic object in bird's view



You can find examples in the folders Examples > CarMakerFunctions > IPGTraffic and Examples > DriverAssistanceSystems.

### 4.8.3 Traffic Object Maneuver Description

Traffic objects can conduct maneuvers which are independent of the test car. The maneuver definition of Traffic objects can be found in the Traffic dialog:

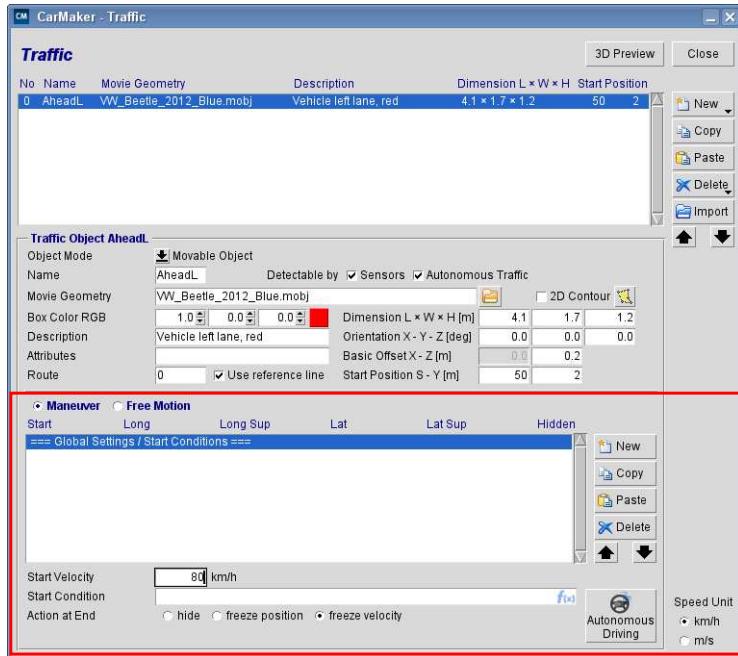


Figure 4.114: Maneuver description for Traffic Objects

First of all, two modes to define a traffic object's maneuvers are available: You can choose between a predefined maneuver list and the free motion mode for the traffic objects.

By selecting *Free Motion*, you can control the traffic object's position via Direct Variable Access (DVA). The positioning information needs to be provided by writing to the quantities:

```

Traffic.<traffic_object_name>.tx
Traffic.<traffic_object_name>.ty
Traffic.<traffic_object_name>.tz
Traffic.<traffic_object_name>.rx
Traffic.<traffic_object_name>.ry
Traffic.<traffic_object_name>.rz

```

For more information regarding DVA, have a look at [section 6.3 'DVA: Online Manipulation of the Simulation' on page 357](#).

By selecting *Maneuver*, the motion of a traffic object can be generally set along the course coordinate. We distinguish between start conditions and maneuver steps that are conducted when the simulation is running (similar to the maneuver definition of the test car). The maneuver steps consist of a nominal, basic motion which can be optionally superimposed by a sinus or triangular motion.

The gross motion of the traffic object is described by the motion of a frame moving along a route with variable velocity. This frame also performs angular motions with a certain angular velocity to maintain its orientation with respect to the street trading.

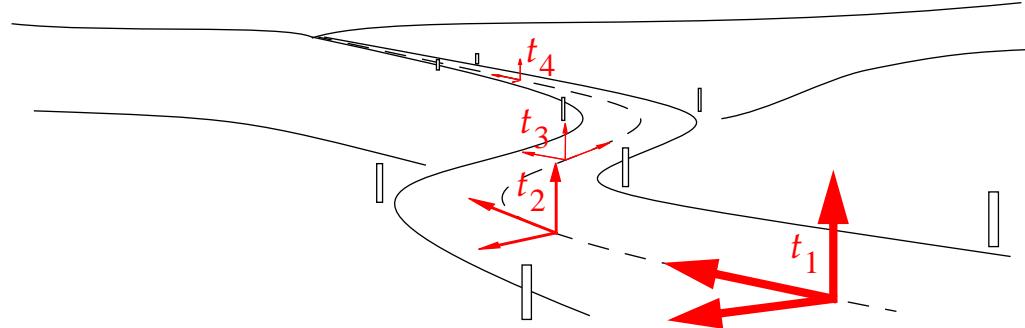


Figure 4.115: Traffic object motion defined in time frames that follow the course coordinates

## Start Conditions

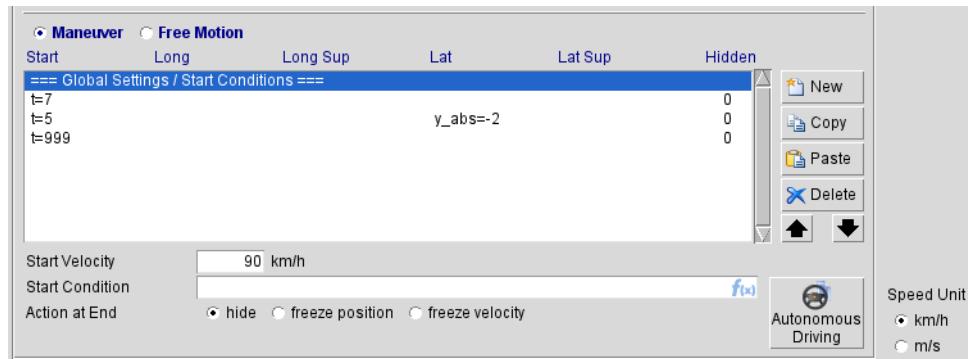


Figure 4.116: Defining start conditions for a traffic object

- Speed Unit** Select in which unit you would like to specify the velocity in the maneuver description of all traffic objects (global setting!).
- Start Velocity (km/h; m/s)** Defines the start velocity of the traffic object. For a parking car or a building, simply enter a value of zero.
- Start Condition** At this point, a start condition for the execution of the maneuver list can be defined using Realtime Expressions (see section '[Realtime Expressions](#)' on page 602).
- Action at End** This option defines the behavior of the traffic object after all minimaneuvers are completed. The traffic object can disappear (*hide*), remain at the position of the final maneuver cycle (*freeze position*) or keep moving with the same, constant velocity of the last maneuver step (*freeze velocity*).

## Maneuver list

A traffic object can perform complex driving maneuvers based on a sequence of *minimaneuvers* - in analogy to the ego vehicle. Use the buttons on the right side of [Figure 4.116](#) to add, copy and paste or delete minimaneuvers. Each minimaneuver description is divided into a longitudinal and lateral motion. The nominal motion in one direction is merely based on the velocity and the relative position of the traffic object to the road center line. These

motions can be extended by superimposed maneuvers in longitudinal and lateral direction. The superimposed longitudinal and lateral behavior can be based on sinus or triangle functions.

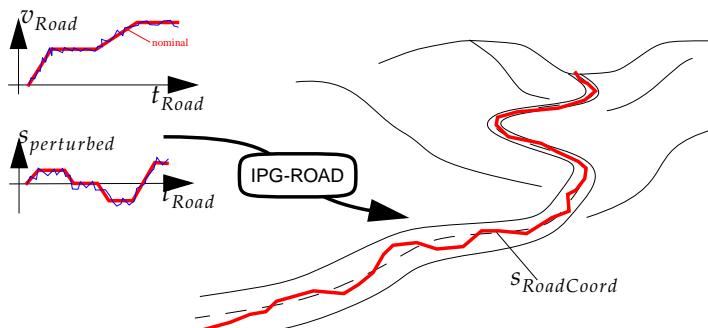


Figure 4.117: Difference between nominal and superimposed motion

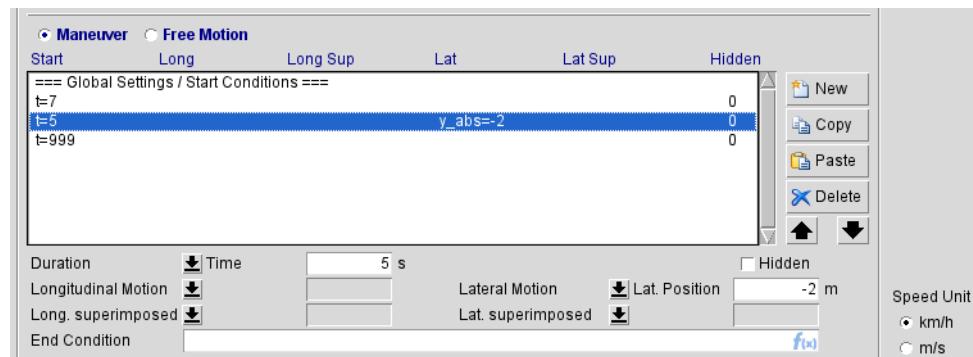


Figure 4.118: Defining minimaneuvers to describe the traffic object's motion

- Duration** Each minimaneuver is characterized by a duration of the maneuver step which defines when the maneuver settings should be achieved. Below, the options available to define the duration of a maneuver step are listed:

Table 4.18: Options to specify the maneuver duration

Parameter	Description
Time [s]	Duration of the maneuver step in seconds
Abs. Time [s]	Maneuver is aborted when the absolute simulation time reaches x seconds
Distance [m]	Distance of the maneuver step in meters
Position [m]	Maneuver lasts until the traffic object reaches the defined absolute road distance in meters

- End Condition** The end condition can abort the maneuver step ahead of time, even if the maneuver target was not yet reached. A dynamic end condition can be specified using Realtime Expressions (see section '[Realtime Expressions](#)' on page 602).

- Longitudinal Motion** The following options to specify the longitudinal motion of the traffic object are available:

Parameter	Description
-	nothing defined

Parameter	Description
Distance [m]	The distance specified should be covered at the end of the maneuver step
Position [m]	The absolute road position in meters should be reached at the end of the maneuver step
Velocity [km/h or m/s]	Target velocity, reached at the end of this maneuver step
Acceleration [m/s <sup>2</sup> ]	The acceleration specified is constantly applied
User Accel.	Longitudinal motion described by a user function in C-code or by DVA (*)
Autonomous	Autonomous driving (see <a href="#">section 4.8.5 'Autonomous Driving'</a> )

(\*) Using the option *User Accel.*, the acceleration can be calculated either by an user implemented function or can be set using Direct Variable Access (DVA).

- User implemented function:

The User.c file, which is located in the src folder of your project directory, may be adjusted in the following way:

```
static double MyTrfObjLongAccFunc (double dt, int id) {
    double LongAcc;
    if (id == x) { //place here the acceleration function for traffic object x
        LongAcc = ...;
    }
    return LongAcc;
}
int User_Register (void)
{
    Set_TrfObjLongAccFunc (MyTrfObjLongAccFunc); //register the user function
    return 0;
}
```

For more information on C-code extensions, refer to the Programmer's Guide.

- Direct Variable Access (DVA):

You can assign any value for the longitudinal motion to the traffic object using the DVA functionality. Either it can be implemented in the Maneuver definition using the Minimaneuver Command Language (see [section B.1.2 'Direct Variable Access Commands' on page 592](#)), or you can directly use the DVA dialog ([section 6.3 'DVA: Online Manipulation of the Simulation' on page 357](#)).

The commands should be used on the quantity called `Traffic.<Name>.LongAcc`.

Setting the acceleration via DVA is only enabled if no user function is registered in the User.c.



**Lateral Motion** At this point, the nominal lateral motion can be specified.

Table 4.19: Definition of the nominal lateral motion

Parameter	Description
-	nothing defined
Lat. Offset [m]	Offset in lateral direction in x meters, measured from the initial road position of the traffic object
Lat. Position [m]	Absolute deviation from the road center line, reached at the end of the maneuver step

Positive values move the traffic object to the left side, negative values to the right.

- Long.  
superimposed** Apart from the nominal longitudinal motion which specifies the road position of the traffic object, its velocity or acceleration, an additional motion can be superimposed. This could be a sinus or triangle, defined by the amplitude and period time - separated by a blank:

Table 4.20: Definition of the nominal longitudinal and lateral motion

Parameter	Description
-	nothing defined
Sinus [m s]	Sinus with amplitude A in [m] and period T in [s], separated by blank
Triangle [m s]	Triangle with amplitude A at period time T, -A at 3*T and 0 at 2*T (5 parameters required, amplitude at 2*T is zero and must not be defined)

- Lat.  
superimposed** Apart from the nominal lateral motion which specifies the lateral road position of the traffic object, an additional motion can be superimposed. This could be a sinus or triangle, defined by the amplitude and period time - separated by a blank. The options are the same as for the longitudinal superimposed motion.
- Hidden** This tick box gives the user the ability to make a traffic object appear and disappear. When the obstacle is invisible, it is not detected by the sensor either. An activated tick box means that the traffic object is visible, which is the default setting.

#### 4.8.4 Animated Pedestrian

An animated pedestrian is defined like any other traffic object. The only difference is the *Movie Geometry* file used. It is based on a special 3D model joints that can be animated to simulate the movements of the whole body. Please refer to [section 7.6 'Pedestrians'](#) on [page 446](#) for more detailed information about the pedestrians models.

The pedestrian model currently available can be found in the product directory under *3DObjects/Characters*.

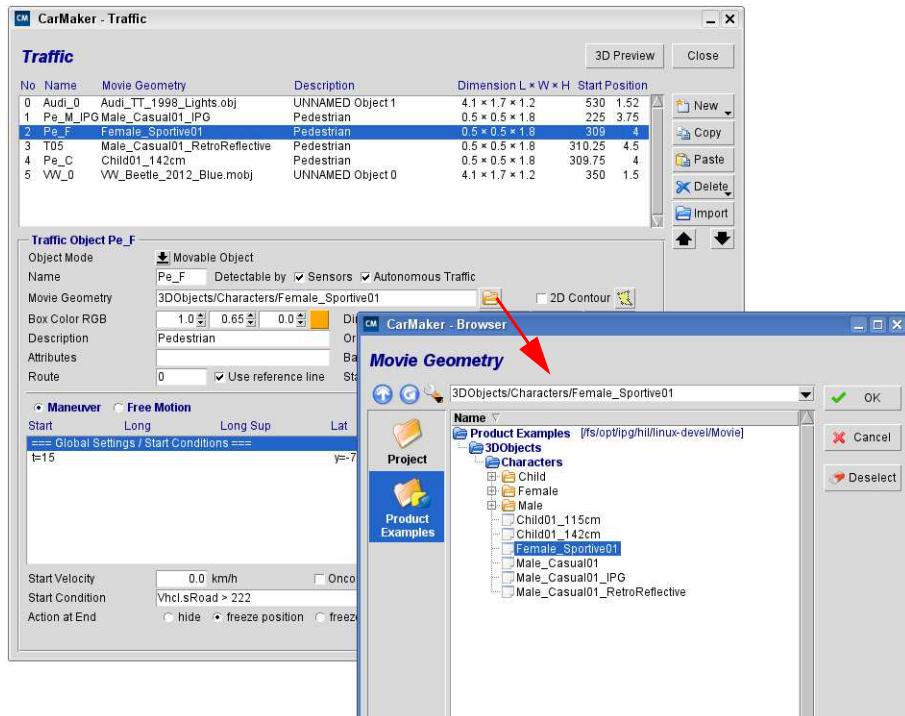


Figure 4.119: Selecting animated pedestrian models



Please find an example TestRun under *Examples/CarMakerFunctions/IPGMovie/Pedestrian*.

## 4.8.5 Autonomous Driving

Using this dialog, which can be opened by selecting the steering wheel icon in the lower right side of the IPGTraffic GUI, you can enable and define autonomous driving for every traffic object separately. The values entered at this point are used only if the option "Autonomous" is selected for the Longitudinal Motion of a traffic object. When a traffic object drives autonomously, it takes into account stop- and speed limit signs.

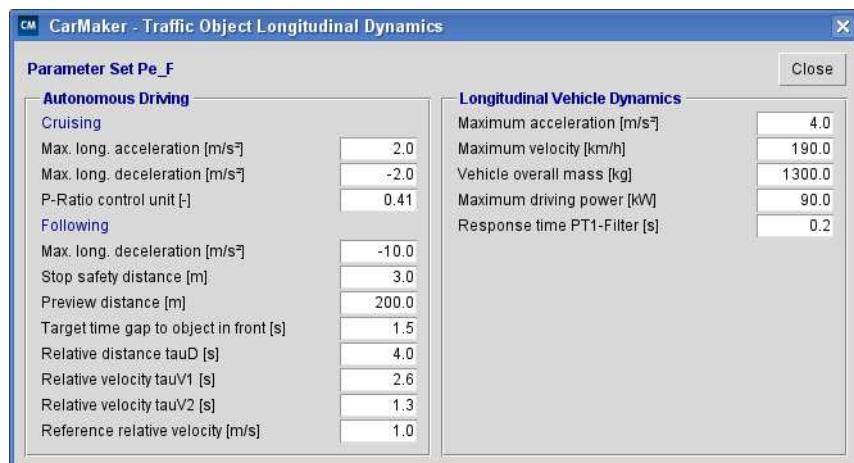


Figure 4.120: Autonomous Driving Dialog

General parameters like the stop safety distance to other traffic objects or the maximum acceleration can be defined in the left part of the autonomous driving dialog, which is divided into the sections *Cruising* and *Following*. If the traffic object is equipped with a following function (works like an ACC sensor), you can also define its sensitivity parameters tauD, tauV1, tauV2 and the reference relative velocity at this point.

TauD is a time constant describing the impact on the control error of the desired time gap or distance respectively. Small values result in a higher control gain and therefore more aggressive ACC control.

TauV1 and tauV2 are the proportional factors of the control loop for the velocity error. Small values result in a more aggressive control but also smaller damping. For this reason, the control is divided in two ranges depending on the current relative velocity between both vehicles. The transition between both ranges is defined by the "reference relative velocity" which is set to 1 m/s by default. For small relative velocities, the gain factor tauV1 shall be high in order to get a damped control behavior. For relative velocities higher than the reference relative velocity, the factor tauV2 shall be small to get a more dynamic control behavior.

To grant a more realistic acceleration of the traffic object, the vehicle parameters defined in the right part of the dialog like mass and driving power are taken into account.



By using the right mouse button in both areas *Autonomous Driving* and *Longitudinal Vehicle Dynamics*, you can select different presets for different driver characteristics and the most common vehicle classes.

## 4.8.6 Output Quantities

With every defined traffic object, a row of output quantities is generated automatically. They can be either plotted in IPGControl or saved to a result file after the simulation was completed. The naming convention of an obstacle's output quantity is quite similar to the parameterization keys. Every quantity starts with the prefix `Traffic`, followed by the traffic object's name which was stated in the Traffic GUI (see [page 129](#)):

```
Traffic.<Name>.*
```

Generated output quantities are e.g. the lateral and longitudinal velocity and acceleration, the road position, relative position and others. For a complete list of all generated output quantities please refer to the Reference Manual.

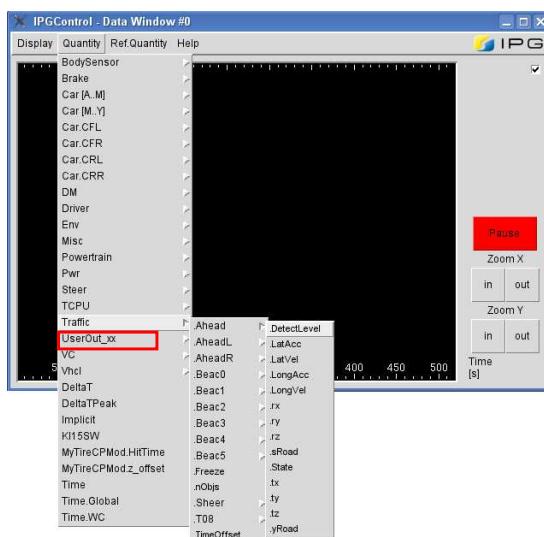


Figure 4.121: User accessible quantities of the traffic objects defined

## 4.8.7 Interaction with Driver Assistance Sensors

With IPGTraffic, the traffic object's absolute motions are calculated. Using this information, the traffic objects can be detected by DASensors placed on the ego vehicle. This interaction between the vehicle sensors and IPGTraffic enables to simulate complex scenarios to evaluate driver assistance systems.

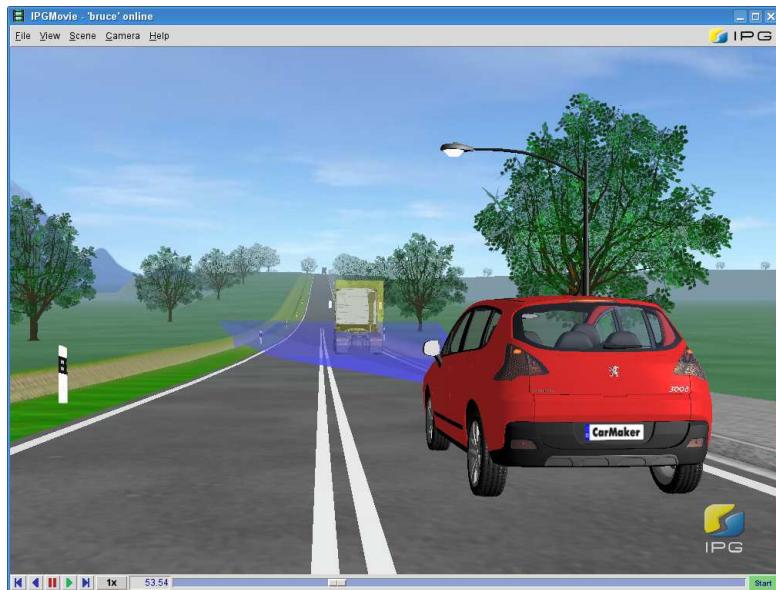


Figure 4.122: Interaction of IPGTraffic and vehicle sensors

For example, a controller for Adaptive Cruise Control (ACC) requires the following input: the distance and direction from the sensor to the traffic object and its relative velocity and acceleration. All these quantities are calculated automatically for each traffic object defined.

Further information about the programming of traffic objects as well as about Driver Assistance Sensors can be found in the Reference Manual.

## 4.9 Environment

The *Environment* module under *Parameters > Environment*, enables to define environmental conditions like the temperature, the time of day or the wind velocity for your simulation. If your model takes these parameters into account, they will influence the results of your simulation.

### 4.9.1 General Parameters

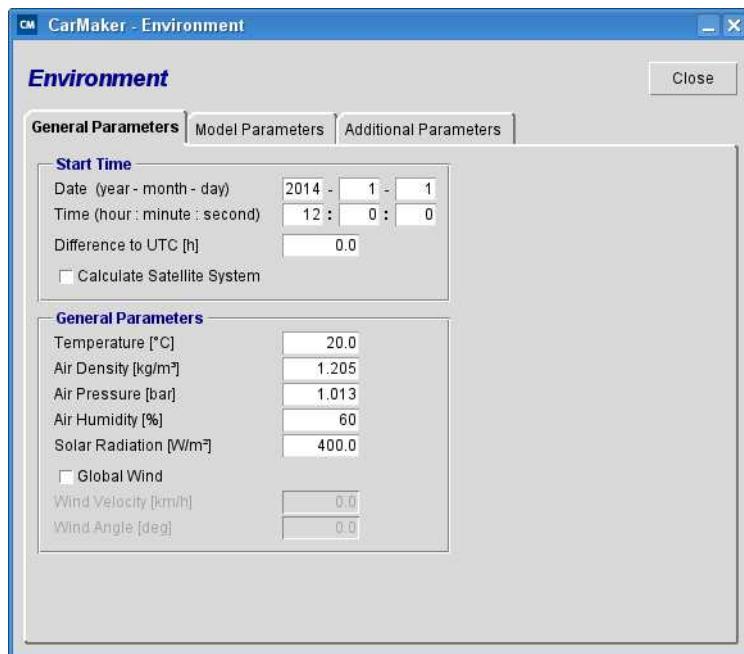


Figure 4.123: Environment - General Parameters

The following parameters can be specified at this point:

- Date** Here, the day which should be simulated can be specified. This parameter is mainly used by the *Global Navigation Satellite System module* (GNSS) to position the satellites in the orbit. The default starting date is January 1st, 2014. The notation is as follows:  
year - month - date



Please note: To simulate the satellite position, you need the navigation message file for that day. The file can be downloaded from <ftp://cddis.gsfc.nasa.gov/pub/gps/data/daily/2015/brdc/> for the year 2015. Only the files ending with "n.Z" are required. The files need to be saved to the *Misc* folder of your CarMaker project directory.

Please find further information about the GNSS module in [section 5.23.8 'Satellite System' on page 294](#) and in the Reference Manual.

- Time** Specifies the starting time of your simulation. The default starting time is 12 p.m.. This parameter is mainly used by the *Global Navigation Satellite System module* (GNSS) to position the satellites in the orbit. The notation is as follows:  
hour : minute : second - month - date

- Calculation Satellite System** This parameter enables the calculation of satellites in the orbit for the *Global Navigation Satellite System module* (GNSS). This module simulates satellites in the orbit. The vehicle model can be equipped with a receiver for the satellite signals.

- Temperature (C)** Specifies the environmental air temperature at mean sea level. The default temperature is 20 degrees.
- Air Density (kg/m<sup>3</sup>)** Specifies the environmental air density at mean sea level. The default air density is 1.205 kg/m<sup>3</sup>.
- Air Pressure (bar)** Specifies the environmental air pressure at mean sea level. The default air pressure is 1.013 bar.
- Air Humidity (%)** Specifies the environmental air humidity in percent. The default air humidity is 60%.
- Solar Radiation (W/m<sup>2</sup>)** Specifies the environmental solar radiation in W/m<sup>2</sup>. The default solar radiation is 400 W/m<sup>2</sup>.
- Global Wind** Selecting this tick box enables a wind, which influences your TestRun. It can be specified using the following parameters:
- Wind velocity (km/h)** Specifies the global constant wind velocity.
- Wind angle (deg)** Specifies the global wind direction angle. Zero degree means a wind direction along the inertial x-axis.

## 4.9.2 Model Parameters

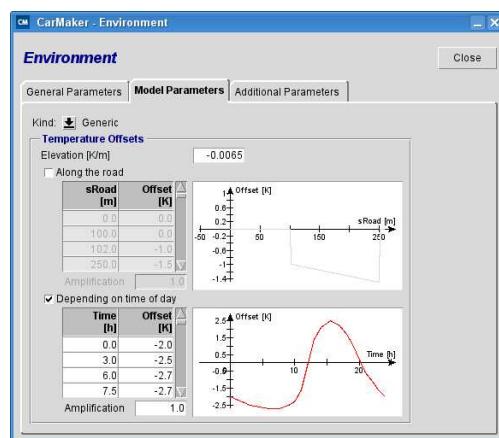


Figure 4.124: Environment - Model Parameters

The *Generic* environment model uses a simplified approach to calculate the environmental air temperature, pressure and density. Other environmental parameters are hold constant.

It is possible to define temperature offsets depending on the time of day and/or driven distance. Both offsets can be defined via the look-up tables.

For detailed information regarding the model parameters, have a look at chapter *Environment Model "Generic"* in the Reference Manual.

Instead of using the *Generic* model, it is also possible to use additional user-defined models by selecting *Kind > Add Model*.

### 4.9.3 Additional Parameters

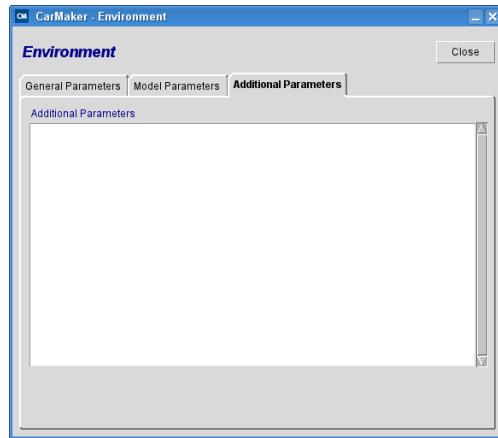


Figure 4.125: Environment - Additional Parameters

Up to now, parameters for user defined models could only be defined within an editor. Via the Environment dialog, it is possible to define them directly using the *Additional Parameters* dialogue.

## Chapter 5

# Parameterization: Vehicle Model

The following subsections describe all parameters required to parameterize the vehicle model using the *Vehicle Editor*. It is reachable in the CarMaker GUI by clicking on *Parameters > Car*. The technical background to each model can be found in the Reference Manual.

The vehicle model uses a separate *Infofile* which is independent on the TestRun description. To save your changes to the vehicle model, it is sufficient to click on *File > Save* in this dialog. Saving in the CarMaker main GUI leads to an update of both, the TestRun and the vehicle *Infofile* (see [section 'Managing the Vehicle Data Set Library'](#)).

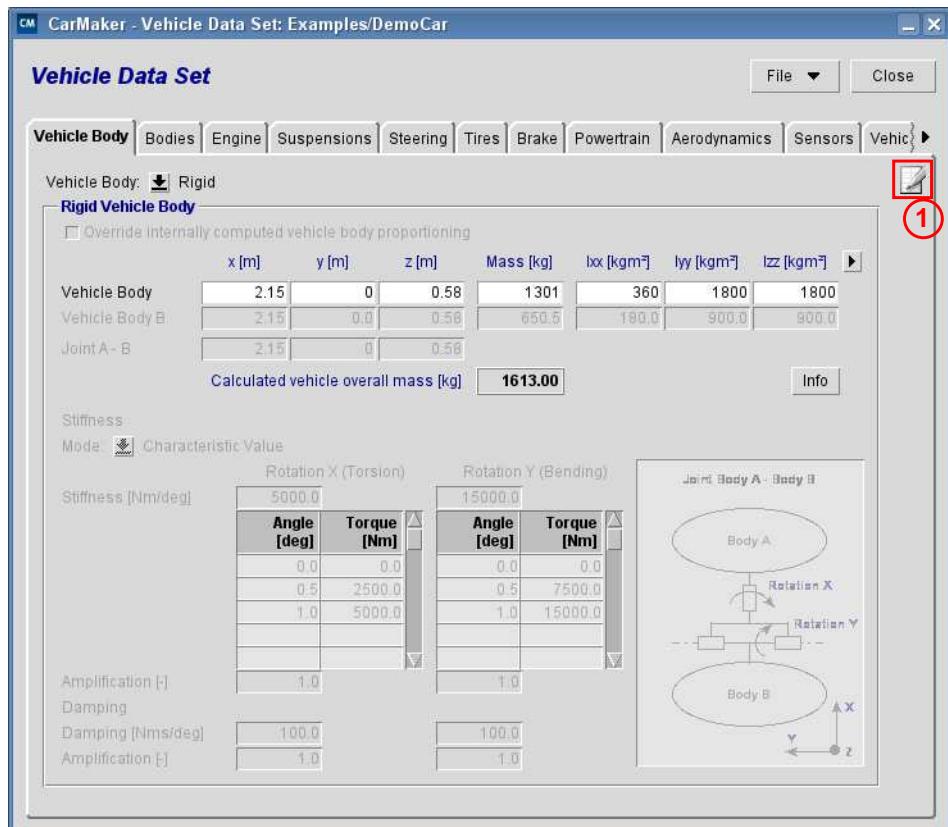


Figure 5.1: The Vehicle Editor

## Tips for the Vehicle Editor



Please note that you can add some comments in each tab. For this, click on the icon in the top right hand corner of the tab (see [Figure 5.1: The Vehicle Editor](#) box 1).

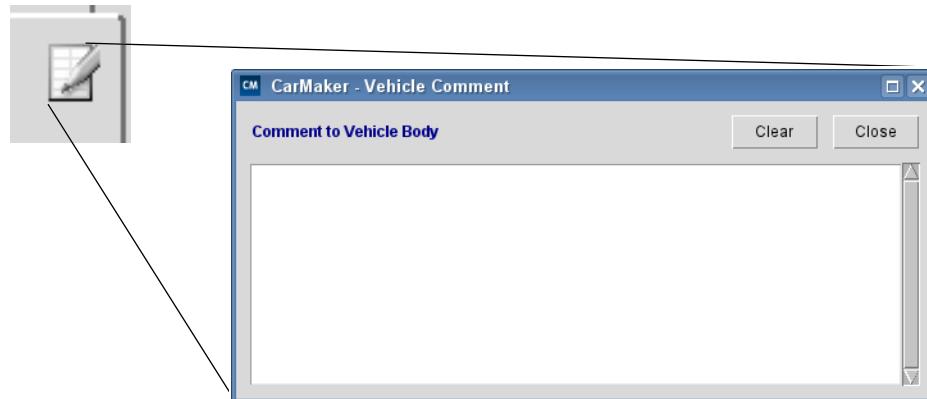


Figure 5.2: Vehicle Comment



In the tables (e.g. Aerodynamics tab or spring stiffness): by clicking only once in a cell you can move to another cell by using the keyboard arrows. If you double-click, you can move to another digit within the cell by using the keyboards arrows and thus edit the cell's entry.

## Managing the Vehicle Data Set Library

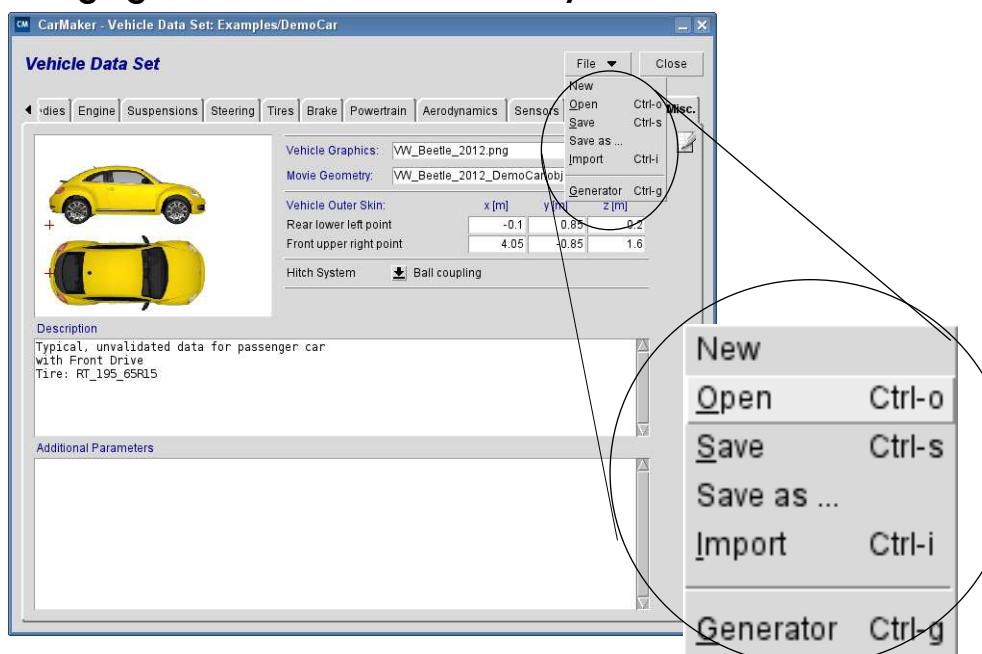


Figure 5.3: Vehicle Data Set Manager

The functionalities that can be found under *File* intend to manage your vehicle data set library. They should already be known before you open this User's Guide, since they have been explained in the Quick Start Guide. If you missed this step, we suggest to refer to the Quick Start Guide, section "Changing the Vehicle and Tire Data" to know more about how to exchange the vehicle data set and both section "Creating a new Vehicle Data Set" and section "Using the Import Feature" to learn how to generate a completely new vehicle data set.

Below you can find a short sum up of the functionalities that are available:

- To use a data set which is already built, you just have to load the corresponding vehicle file via the vehicle editor. Click on *File > Open*.
- You can also create a new vehicle data set from scratch. In this case, be aware that even if you choose this option, the vehicle editor is filled with default values (at the exception of the vehicle picture). If you did not finish the vehicle parameterization before closing, do not forget to add a comment where you stopped (field *Description* in the tab *Misc.*). Click on *File > New*.
- Once you have parameterized your vehicle data set, you can save your changes or alternatively save them to a new file. Click on *File > Save* or click on *File > Save As*.
- Note that you have the possibility to import a part of the vehicle from another vehicle data set. Click on *File > Import*.
- The Generator gives you the possibility to generate a vehicle model with basic data only.

## 5.1 The Vehicle Data Set Generator

You can also generate a vehicle with very basic data. Of course, the generated vehicle is not validated, but it corresponds to a vehicle of a certain class. This can be extremely useful if you work on a new regulation system and are not interested in the exact vehicle parameterization itself. In the Vehicle Data Set Editor click on *File > Generator*.

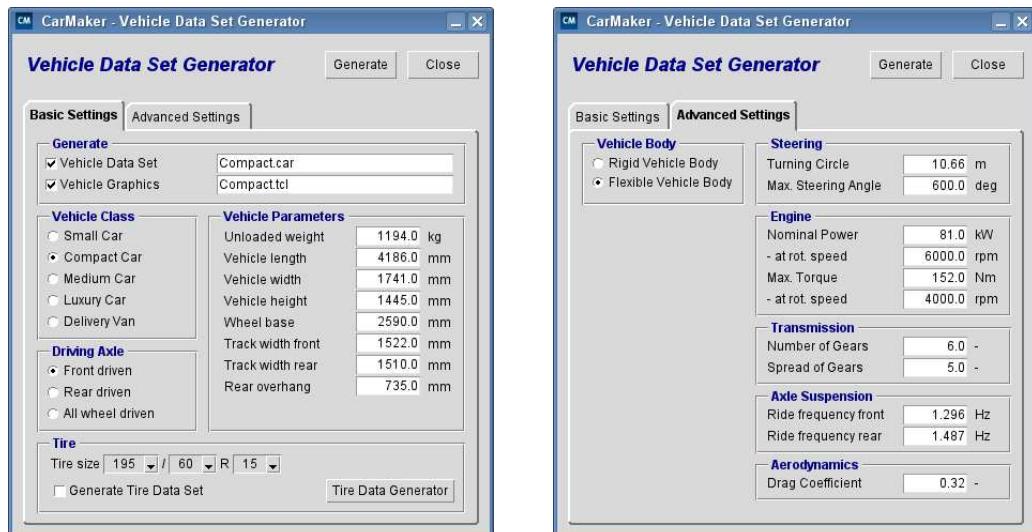


Figure 5.4: Dialog of the Vehicle Data Set Generator

**Vehicle Data Set** At this point, you can define a name for the generated vehicle.

**Vehicle Graphics** If this field is activated, a picture file is generated which suits to the vehicle kind you chose. The picture file is a tcl file, saved to the *Data/Pic* folder of your project directory, named as your vehicle data set. The picture is displayed in the CarMaker main GUI as well as on the *Misc.* tab of the vehicle data set editor.

**Vehicle Class** By selecting the vehicle class, you influence the basic attributes of the vehicle you generate. Just by selecting a vehicle type, all data required to simulate a car of this class are available. There is no other information required to generate a data set - the other parameters are optional to refine the vehicle model according to your needs.

<b>Driving Axle</b>	At this section you can select the powertrain kind. Available options: front driven, rear driven, all wheel driven.
<b>Vehicle Parameters</b>	At this point you have the possibility to specify some general parameters such as the unloaded weight, weight distribution, outer dimensions and wheel size. The rear overhang is the distance from the rear axle to the rearmost point on the body.
<b>Tire</b>	At this point, the desired tire size must be selected. If no suitable tire model is available, select the tick box <i>Generate Tire Data Set</i> . By clicking the button <i>Tire Data Set Generator</i> afterwards, the respective window will be opened directly. For more information regarding the <i>Tire Data Set Generator</i> , have a look at <a href="#">5.27.3Tire Data Set Editor</a> .
	The <i>Advanced Settings</i> offer even more parameters to generate your vehicle data more detailed.
<b>Main Body</b>	You can use a rigid or a flexible body frame. Please note that the rigid model should be sufficient for most applications. The simulation results can be distorted by selecting a flexible body frame without having detailed measurements of your vehicle.
<b>Steering</b>	The configuration of the steering system can be optimized by defining the turning circle diameter and the maximum steering angle.
<b>Engine</b>	To enhance the engine characteristics of the generated vehicle, the nominal and maximal power and the corresponding engine speed can be configured.
<b>Gear Box</b>	The number of gears and their spreading can be adapted in this box.
<b>Aerodynamics</b>	Here you can adjust the aerodynamical drag coefficient. Once you finished the parameterization of the Vehicle Data Set Generator according to your requirement of accuracy, you can generate the vehicle data set by clicking on the <i>Generate</i> button.
<b>Axle Suspension</b>	The frequency of the front and rear springs and dampers can be adjusted in this box. The higher the frequency, the stiffer the ride.   Please note that you need to save the generated vehicle data set in the vehicle data set editor under <i>File &gt; Save</i> . This data set will be permanently available in the selection.  If you would like to optimize your vehicle data set further or if you would like to create a very detailed new vehicle data set, you can use the vehicle data set editor. Please find an elaborate description of the submodels on the following pages.

## 5.2 CarMaker Coordinate Systems

In CarMaker, coordinate systems - called frames - with different origins are defined. However, the direction of the axes is always the same, following DIN 70000:

- x-axis: Points in driving direction of the vehicle
- y-axis: Points 90 deg to the left side of the x axis
- z-axis: Points upwards vertically

An overview of all frames available can be found in the Reference Manual, [section 5.2 'CarMaker Coordinate Systems'](#). The reference frame for parameterization of the vehicle data set is *Fr1 (Frame One)*. Its origin is located at the rearmost point of the vehicle, on the ground.



In case that definition does not fit to the position of the origin your input data refers to, there is the possibility to define a vector which creates a new frame, called *FrD* (*Design Frame*). All coordinates you enter to the vehicle data set then refer to *FrD* instead of *Fr1*. The coordinate translation is done internally by CarMaker (see [page 154](#)).

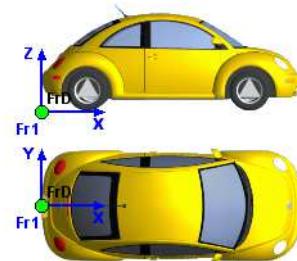


Figure 5.5: Definition of *Fr1*, used for vehicle parameterization

## 5.3 Bodies

The masses of the vehicle models are described in the tabs *Vehicle Body* and *Bodies*

The screenshot shows two instances of the CarMaker Vehicle Data Set editor window.

**Vehicle Body Tab:**

Vehicle Body	x [m]	y [m]	z [m]	Mass [kg]	Ixx [kgm <sup>2</sup> ]	Iyy [kgm <sup>2</sup> ]	Izz [kgm <sup>2</sup> ]
Vehicle Body	2.15	0	0.58	1301	360	1800	1800
Vehicle Body B	2.15	0.0	0.58	650.5	180.0	900.0	900.0
Joint A - B	2.15	0	0.58				
				Calculated vehicle overall mass [kg]	<b>1613.00</b>		

**Stiffness Tab:**

Stiffness [Nm/deg]	Rotation X (Torsion)		Rotation Y (Bending)	
	Angle [deg]	Torque [Nm]	Angle [deg]	Torque [Nm]
0.0	0.0	0.0	0.0	
0.5	2500.0	0.5	7500.0	
1.0	5000.0	1.0	15000.0	

Amplification [-]: 1.0      Damping [Nms/deg]: 100.0      Amplification [-]: 1.0

**Bodies Tab:**

Body	x [m]	y [m]	z [m]	Mass [kg]	Ixx [kgm <sup>2</sup> ]	Iyy [kgm <sup>2</sup> ]	Izz [kgm <sup>2</sup> ]
Wheel Carrier FL	3.258	0.754	0.298	18.0	0.2	0.2	0.2
Wheel Carrier FR	3.258	-0.754	0.298	18.0	0.2	0.2	0.2
Wheel Carrier RL	0.73	0.747	0.293	13.0	0.1	0.1	0.1
Wheel Carrier RR	0.73	-0.747	0.293	13.0	0.1	0.1	0.1
Wheel FL	3.258	0.754	0.298	25.0	0.4	1.2	0.4
Wheel FR	3.258	-0.754	0.298	25.0	0.4	1.2	0.4
Wheel RL	0.73	0.747	0.293	25.0	0.45	0.7	0.45
Wheel RR	0.73	-0.747	0.293	25.0	0.45	0.7	0.45
Number of Trim Loads:	0				Mounting		

**Position Tab:**

Position	x [m]	y [m]	z [m]
Origin Fr1	0.0	0.0	0.0
Aero Marker	3.65	0.0	0.6
Hitch	0.0	0.0	0.4
Jack FL	2.90	0.754	0.298
Jack FR	2.90	-0.754	0.298
Jack RL	1.10	0.747	0.293
Jack RR	1.10	-0.747	0.293

Legend: ● Origin Fr1   ● Positions   ● Geometry Bodies   ● Geometry Trim Loads

**Diagrams:**

- Flexible Vehicle Body:** A schematic diagram showing two bodies, Body A and Body B, connected at a joint. The joint allows rotation around both the X and Y axes.
- Vehicle Body:** A 3D perspective view of a yellow Volkswagen Beetle car with a coordinate system (X, Y, Z) and a 2D top-down view.

Figure 5.6: Definition of the main body and sub bodies in the Vehicle Data Set editor

### 5.3.1 Model Kinds

Please find detailed information about the technical background in the Reference Manual, sections "Vehicle Body" and "Vehicle Model 'Flexibe'".

**Vehicle Body** In the tab *Vehicle Body*.

The Parameter *Kind* enables you to choose between the following models:

Table 5.1: Vehicle Body Models

Model Kind	Description
Rigid	Vehicle body consists of one rigid mass.
Flexible	Vehicle body is divided into two masses connected by a rotational joint with parameterizable stiffness.

The flexible body model regards bending and torsion of the body frame.

If you do not have very good values of the vehicle stiffness, it is recommendable to use the model *Rigid Body*. Indeed, the flexibility of the body will have a much too high influence that you will not be able to handle, and subsequently producing unusable results. Switching off the flexibility of the vehicle will lower the importance of effects that you cannot evaluate. In addition to that, many successful simulation results and validations have been performed with this vehicle model before CarMaker 2.2, proving that the model *Rigid Body* already is a very good model.

However, if accurate values of your body stiffness are available, you should use the *Flexible Body* model. In this case you are sure, that the flexibility has only the correct effects on the vehicle behavior. This model particular is recommended for competition purposes or fine suspension tuning.



You can find examples in the folder Examples > CarMakerFunctions > FlexBody

### 5.3.2 Mass and Inertia Definitions

According to the selected vehicle model (flexible or rigid), a different number of bodies can be parameterized. In case of a flexible body mode, the main body is split into two bodies. The main body / bodies include all sprung masses of the vehicle.

The unsprung masses are defined in the *Bodies* tab under *Wheel* and *Wheel Carrier*. Heavy loads can be extracted of the main body and defined separately using trim loads.

Note: All masses of the *Bodies* tab are added to the main body's mass internally.



The following table gives an overview of the masses available under respect of the vehicle model (*Rigid Body* or *Flexible Body*) selected:

Table 5.2: Definition of the main bodies in the *Vehicle Body* tab

Name of the Body	Model Kind	Comments	Tab
Vehicle Body A	Rigid Body	Includes all sprung masses, optionally with the exception of the engine (see <i>Engine A</i> ) and user defined trim loads.	Vehicle Body
	Flexible Body	Includes all sprung masses at the front, optionally with the exception of the engine (see <i>Engine A</i> ) and user defined trim loads.	Vehicle Body

Table 5.2: Definition of the main bodies in the *Vehicle Body* tab

Name of the Body	Model Kind	Comments	Tab
Vehicle Body B	Flexible Body	Includes all sprung masses at the <u>rear</u> , optionally with the exception of the engine (see “Engine B”). This body is optional. You may automatically distribute the mass on both body A and B: see description of the <i>Joint Mode</i> in the section ‘ <i>Joint Mode</i> ’ on page 153	Vehicle Body

Table 5.3: Definition of the extra masses in the *Bodies* tab

Name of the Body	Model Kind	Comments	Tab
Wheel Carrier (x4)	Rigid Body and Flexible Body	Unsprung but non-rotating masses (wheel carriers, brake callipers, rods or other fixed elements whose weight pushes on the wheel carrier). If an element is only half unsprung (e.g. steering rod), only the half of its mass should be regarded here.	Bodies
Wheel (x4)	Rigid Body and Flexible Body	Unsprung and spinning masses (wheels, disk brakes). The position of the wheels is defined assuming that the tires are not compressed.	Bodies
Trim Loads	Rigid Body and Flexible Body	Optional. Add numerous extra loads to simulate the effect of passengers, a full tank or boot etc. Select <i>Fr1A</i> to add load to the front part of the flexible main body and <i>Fr1B</i> to add load to the rear part .	Bodies

x, y, z (m)  
Mass (kg)  
 $I_{xx}, I_{yy}, I_{zz}$  ( $\text{kg}\cdot\text{m}^2$ )

Each of the masses has the following properties:

- Position of the center of gravity of the main body in the three fields x, y and z of the *Vehicle Body* tab.  
The location of each body, especially of the main body, is parameterized in the design state. The design state can be a virtual state (possibly even be a state where the springs are not compressed), or the static state. The position of the wheels is defined assuming that the tires are not compressed either.  
Note that the state in which you parameterize the masses' positions has an influence on the parameterization of the kinematics. For this, read the [section 5.3.4 'Coordinate System Fr1 and FrD for bodies' on page 154](#).
- Under *Bodies* define the weight influence of the body.
- Under  $I_{xx}$ ,  $I_{yy}$  and  $I_{zz}$  the moments of inertia of this body mass can be defined in the axis orientation of frame1. The position of the center of gravity of the mass regarded is taken as origin.
- By clicking on the small arrow that is at the end of the line (near  $I_{yy}$  [ $\text{kg}\cdot\text{m}^2$ ]), you have access to the products on inertia  $I_{yz}$ ,  $I_{xz}$  an  $I_{xy}$ . The products of inertia are of lower importance, they can be left empty in a first approximation.

Overall mass

The field calculated overall mass in the main body tab states the current weight of the whole vehicle including all masses defined separately in the *Bodies* tab.

### 5.3.3 Flexible Body Mode

#### Joint Properties

**Joint Mode** In the tab *Vehicle Body*.

If you have chosen the model *Flexible Body*, you also have to parameterize the joint between the front and rear bodies (A- and B-Bodies). The interface offers you to parameterize only the global weight of the body, or the weight of each body A and B separately (by activating *Override internally computed vehicle body proportioning*):

Table 5.4: Weight Distribution on the Flexible Vehicle Body Model

Mode	Description
Body A	The mass and inertia of the main bodies are in fact parameterized only once (in the field <i>Vehicle Body A</i> ) and then automatically and equally distributed between both bodies (a half to the part A and a half to the part B). The new positions of the masses are chosen so that the total inertia, mass and CoG stay unchanged.
Body A+B	The mass and inertia of the main bodies are parameterized separately for each of the bodies A and B (in the fields <i>Vehicle Body A</i> and <i>Vehicle Body B</i> ).

**Joint A-B (m)** Here you can define the position of the joint between both bodies. Only available when *Override internally computed vehicle body proportioning* is selected.

The coordinate system is the same as for the positioning of the bodies. If you do not have this kind of information, you can approximate this point to the middle of the wheel base or activate the internally computed proportioning.

**Stiffness/Damping Mode** The stiffness of the joint can be either linear (enter a single parameter) or non-linear (using a table of values):

Table 5.5: Stiffness/Damping Modes for the flexible Vehicle Body

Mode	Description
Characteristic Value	The torsional stiffness of the vehicle is parameterized by linear coefficients (unit: Nm/deg).
Look-Up Table 1D	The torsional stiffness of the vehicle is parameterized by non-linear characteristics (table form: Nm vs. deg).

The torsional stiffness has to be parameterized in two directions: X (torsion) and Y (bending). Typically those values can be gained from real measurements.

For most passenger cars the bending stiffness is of lower importance and a quite high value can be left at this place. But if you are simulating a convertible, a light truck or any kind of long vehicle with lower dynamic requirements (light or heavy commercial vehicles, vans, ...), the bending stiffness should not be neglected.

**Stiffness Amplification (-)** This parameter enables to modify the stiffness very fast and easily, either to compare two simulation results or to convert the values to the units required by CarMaker. Usually leave the value to 1.0.

**Damping (Nm.s/deg)** For the damping, a single linear coefficient is necessary for each degree of freedom. This value is quite difficult to obtain and has a lower effect than the stiffness. The default value of 100 Nms/deg should be fine for most common passenger vehicles.

<b>Damping Amplification (-)</b>	This is a parameter that enables to modify the damping very fast and easily, either to compare two simulation results or to convert the values to the units required by CarMaker. Usually leave the value to 1.0.
----------------------------------	---

### 5.3.4 Coordinate System Fr1 and FrD for bodies

**Origin Fr1: the Coordinate System (m)**

In the tab *Bodies*.

To parameterize all necessary positions, CarMaker uses a coordinate system called *Fr1* which is linked to the vehicle body. The origin of this coordinate system has to be placed behind the rear wheels, e.g. the x-values of the wheels all have to be positive (see [section 5.2 'CarMaker Coordinate Systems' on page 148](#) of the User's Guide and [section 1.2 'Car-Maker Axis Systems'](#) of the Reference Manual).

However, there is the possibility to define a design frame *FrD*, all coordinates relate to instead. Internally, CarMaker transfers all coordinates to fit in *Fr1*. The origin of *FrD* is defined by the vector *Origin Fr1* which points from the origin of *FrD* to *Fr1*.

The example in [Figure 5.7](#) shows a design frame which was translated by -0.5m, 0.0m and +0.5m in x direction compared to frame *Fr1*.

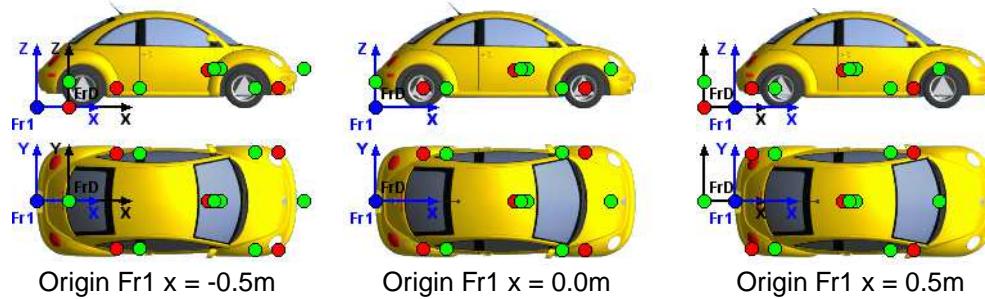


Figure 5.7: Moving the design frame's origin

### 5.3.5 Other Parameters

**x, y, z (m)**

In the tab *Bodies*.

- *Aero Marker*: This parameter belongs to the aerodynamic parameterization. See the [section 5.22 'Aerodynamics' on page 280](#).
- *Hitch*: This is the position of the hitch. This parameter is used only if you parameterize a trailer.
- *Jacks*: When you apply a force on a jack (for instance through DVA), the vehicle body is lifted. This functionality is used for air springs ECU testing.
- *Picture*:

By bringing the cursor on the colored points symbolizing the position of the bodies, you can display the coordinates of those bodies.

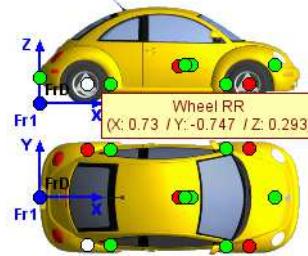


Figure 5.8: Body Coordinates

The picture that is displayed in the vehicle editor can be chosen in the tab *Misc.* of the vehicle data set editor. You can define your own picture, too. Please refer to [section 5.25 'Miscellaneous' on page 299](#).

## 5.4 Loads and Trim Loads

In the following you will find a few information about the additional loads.

As you have seen, with the so called *Trim Loads* there is the possibility to add some extra loads in order to simulate passengers, luggage, heavy measurement equipment and so on. If you check the static position of the vehicle using ModelCheck, the effects of trim loads will be included e.g. to the height of the vehicle center of gravity or the wheel positions in the start off configuration.

For some specific simulation purposes, you may want to simulate a TestRun with an additional load on the vehicle, but you still want to check the static state of the vehicle without it. This is also possible with the so called *Loads* (not to be confused with the *Trim Loads!*). The *Loads* are saved in the TestRun file, whereas the *Trim Loads* are saved in the vehicle data set. *Loads* can be specified in the CarMaker main GUI, under *Parameters > Loads*:

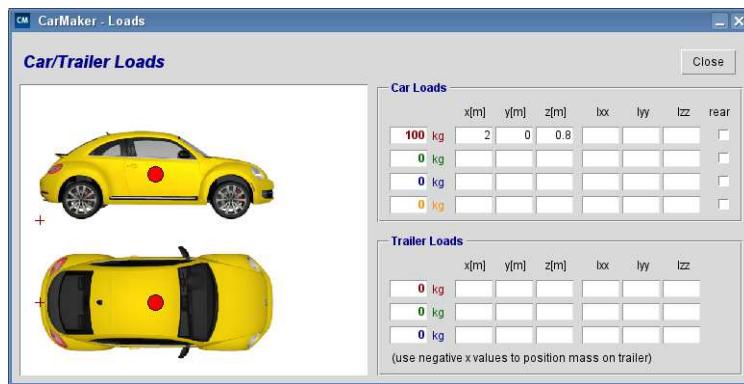


Figure 5.9: Definition of Loads

Contrary to the *Trim Loads*, the *Loads* can be used for the vehicle as well as for the trailer if one is present. However, the number of *Loads* is limited to 4 for the vehicle, and 3 for the trailer. By selecting the checkbox *rear*, it is possible to define loads for the rear part of a flexible car (two loads for one body). Another point is that the *Loads* are not considered during a ModelCheck analysis.

But you may ask: How should I decide to use the *Loads* or the *Trim Loads*? Basically, you should use the *Trim Loads*. Indeed the number of *Trim Loads* is unlimited, and you can parameterize them in the same window as the other bodies. The *Loads* are interesting in the following cases:

- loads should be added to a trailer
- simulate a TestRun with a load, but check the static state without it.

## 5.5 Engine Mount / Body

CarMaker offers the possibility to simulate an elastically mounted engine. By activating this feature, an additional mass is added to the model which needs to be connected to the main body by a joint. The generated engine torque is then supported by the body through this joint. The required settings can be specified in the tab *Engine* of the Vehicle Data Set Editor:

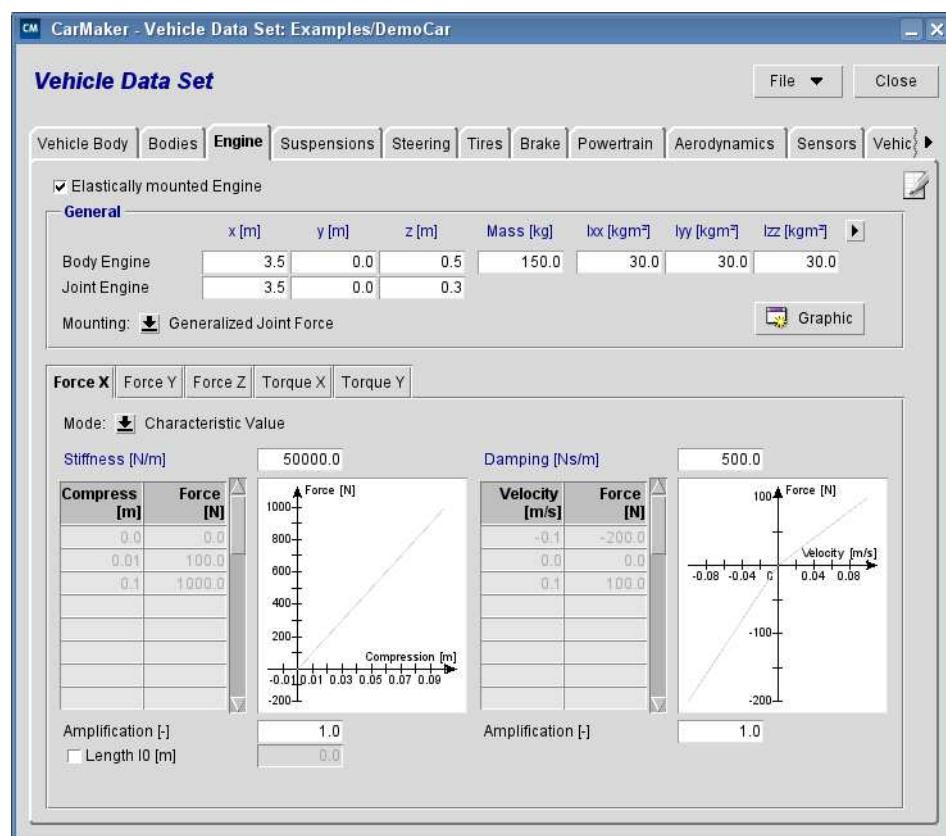


Figure 5.10: Parameterization of the elastically mounted engine

**Body Engine** The engine body is parameterized similarly to the vehicle body. The position of the center of gravity in the *Design Frame* is required, as well as the moments and products of inertia if known.

**Joint Engine** Here the engine joint - the connection point from the engine mass to the vehicle body - is specified.

To check the engine body and joint position, use the *Graphic* button.





CarMaker offers the possibility to integrate the complete Sherpa Engine thermodynamic model library. To get more information about this, have a look at our homepage or contact our Service Team.

**Mounting** The kind of mounting suspension can be selected. The following options are available:

Kind	Description
Generalized Joint Force	All force elements apply at the joint. The generalized forces are: force in z-direction, torque around x and y axis. This is the default mode.
3 explicit Force Elements	Three freely located mounting points with force elements.
4 explicit Force Elements	Four freely located mounting points with force elements.

Depending on the mounting kind selected, the force elements need to be specified by the following parameters:

### Mounting: Generalized Joint Force

**Mode** Each of the five force elements Force X, Force Y, Force Z, Torque X and Torque Y is made of a spring and damper element. To characterize the elements, several modes are available:

Mode	Description
Characteristic Value	Linear spring and damper characteristic defined by a single coefficient in [N/m] for the spring stiffness and in [Ns/m] for the damping.
1D Look-Up Table	Non-linear spring and damper characteristic defined by a look up table. For the spring, the map requires the force in [N] over compression in [m], the damper element is specified by force in [N] over velocity in [m/s].
DVA	The desired force element force is set using the Direct Variable Access on the quantities <i>Car.Eng.GenFrc.z</i> , <i>Car.Eng.GenTrq.x</i> , <i>Car.Eng.GenTrq.y</i> . Please find further information about DVA in section 6.3 'DVA: Online Manipulation of the Simulation' on page 357. The stiffness value is used in the preparation phase (calculation of static equilibrium).
User Function	The desired force element force is calculated by an user implemented function based on a CarMaker c-code extension. Please find an example in the Reference Manual.

**Amplification (-)** The *Amplification* factor can be used to easily manipulate the spring and damper characteristics by a multiplication factor given in this field.

- Length I0  
(m)** Optionally, the free, unstretched length of the spring and damper element can be specified.

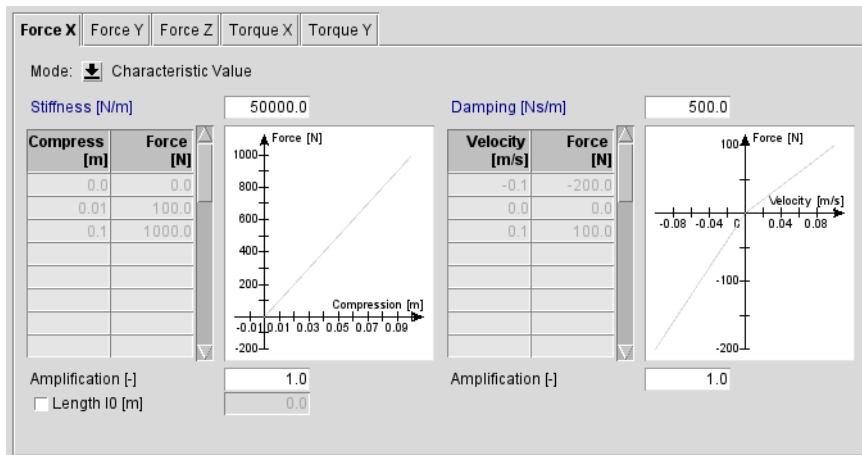


Figure 5.11: Parameterization of the force elements in mounting mode *Generalized Joint Force*

## Mounting: 3/4 explicit force elements

- Mode** Again, each of the max. four force elements is made of a spring and damper element. To characterize the elements, the following modes are available:

Mode	Description
Characteristic Value	Linear spring and damper characteristic defined by a single coefficient in [N/m] for the spring stiffness and in [Ns/m] for the damping.
1D Look-Up Table	Non-linear spring and damper characteristic defined by a look up table. For the spring, the map requires the force in [N] over compression in [m], the damper element is specified by force in [N] over velocity in [m/s].
DVA	The desired force element force is set using the Direct Variable Access on the quantities <i>Car.Eng.Frc.0.Frc</i> ... <i>Car.Eng.Frc.4.Frc</i> . Please find further information about DVA in section 6.3 'DVA: Online Manipulation of the Simulation' on page 357.
User Function	The desired force element force is calculated by an user implemented function based on a CarMaker C-code extension. Please find an example in the Reference Manual.

- Amplification  
(-)** The *Amplification* factor can be used to easily manipulate the spring and damper characteristics by a multiplication factor given in this field.

- Length I0  
(m)** Optionally, the free, unstretched length of the spring and damper element can be specified.

**Position X - Y - Z (m)** Here, the position of the force element is defined in *FrD*.

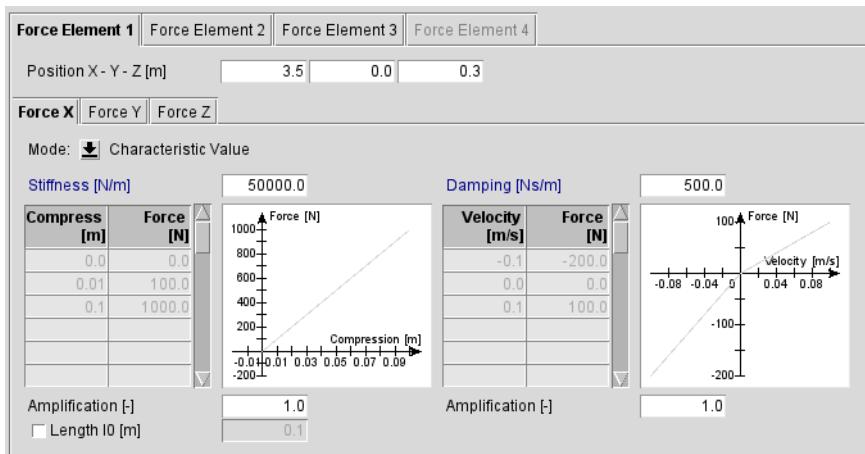


Figure 5.12: Parameterization of the force elements in mounting mode *Explicit Force Elements*

## 5.6 Suspension: Spring

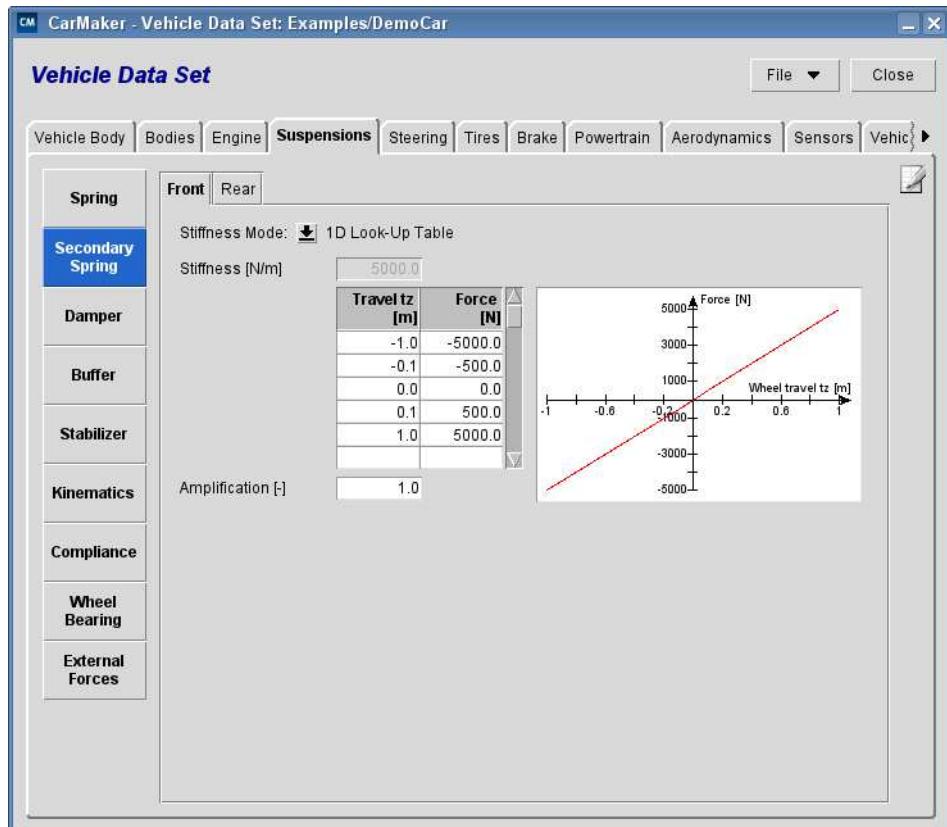


Figure 5.13: Definition of the spring characteristics

For a given axle, the springs characteristics are the same for both sides of the vehicle. The spring is modeled as a component that generates a force when it is compressed/stretched. The spring is seen here as a "stand alone" component which properties (free length and stiffness) can be measured independently.

This means that you do not have to specify the geometrical properties. Its length variation in function of the wheel travel is defined in the kinematics section (see [section 5.11 'Suspension: Kinematics' on page 166](#))

For additional information, also see the Reference Manual, section "Springs".

**Mode** For each front and rear axle, the spring stiffness can either be defined with a simple coefficient (mode *Characteristic Value*) if the spring force is a linear function of the spring length variation, or in a table (mode *Look-Up Table 1D*) otherwise.

**Stiffness  
(N/m)** This parameter either is a simple coefficient or a table according to the selected parameter *Kind*. A positive spring length variation means that the spring is compressed.

If the application requires it, you may also parameterize the stiffness for the traction domain (negative length variation). If undefined, CarMaker estimates the characteristics in this domain by using the first value given and keeping it constant.

**Amplification  
(-)** This parameter enables to scale the spring stiffness very quickly for test purposes instead of modifying the whole table. You can also use this parameter to convert the values to fit to the units required by CarMaker. The default value is 1.0.

**Length  $l_0$   
(m)** This is the relaxed or unstretched length of the spring. The resulting spring force depends on the difference between the relaxed length  $l_0$  and the length  $l$ . The length is the current distance between the lower and upper attachment point of the spring. Deactivating the checkbox lets CarMaker calculate the length of the unstretched spring.



Before starting a simulation with a new vehicle data set, you should check if the value for the relaxed spring length is correct, or if it needs to be tuned. For this, you can compare the parameter *spring coord* in the equilibrium state output of the Model Check. For more information please refer to the Reference Manual [section 6.4 'Model Check' on page 360](#).

## 5.7 Suspension: Secondary Spring

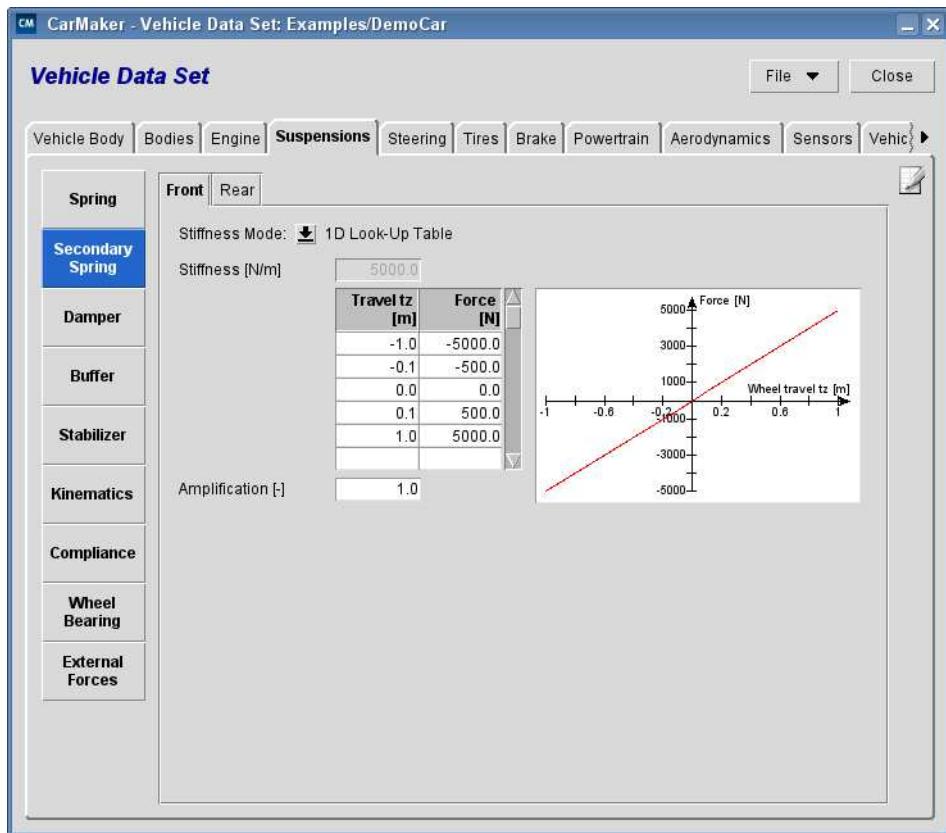


Figure 5.14: Definition of the secondary spring characteristics

- Mode** For each front and rear axle, the stiffness of the secondary spring can either be defined with a simple coefficient (mode *Characteristic Value*) if the secondary spring force is a linear function of the wheel travel variation, or in a table (mode *Look-Up Table 1D*) otherwise. If the mode *-not specified-* is selected, the secondary spring is deactivated.
- Stiffness (N/m)** This parameter either is a simple coefficient or a table according to the selected parameter *Mode*. A positive wheel travel *tz* means that the wheel center is lifted up and that it generates a positive secondary spring force.
- Amplification (-)** This parameter enables to scale the stiffness of the secondary spring very quickly for test purposes, instead of modifying the whole table. You can also use this parameter to convert the values to fit to the units required by CarMaker. The default value is 1.0. If the value is 0.0, the secondary spring is deactivated.

## 5.8 Suspension: Damper

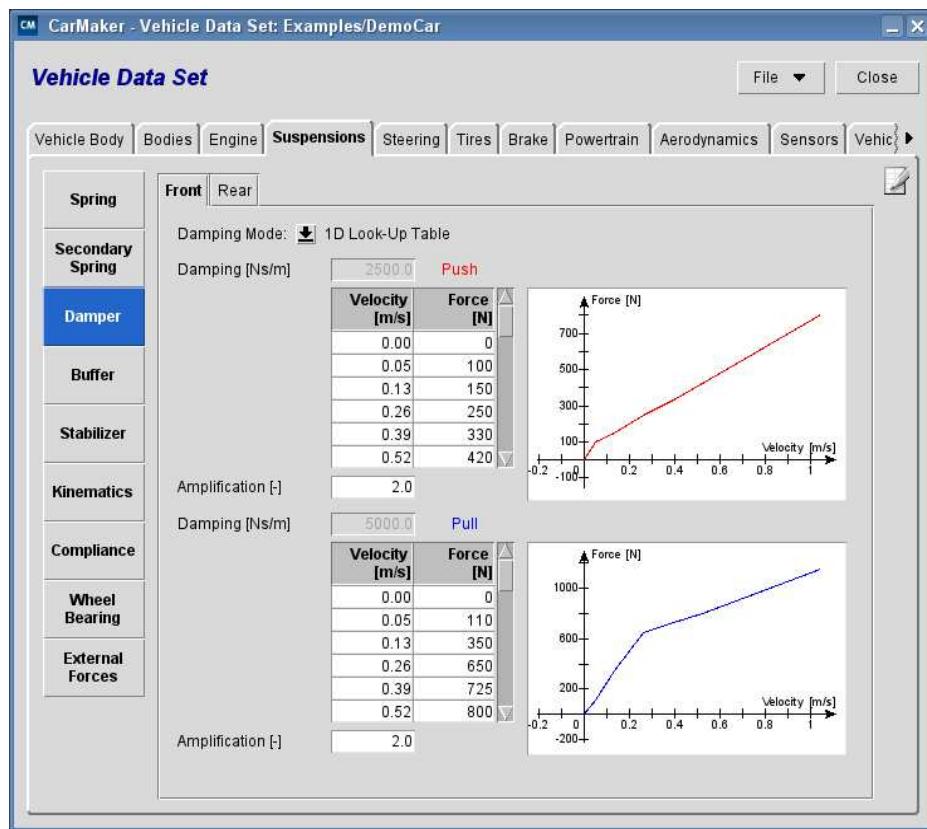


Figure 5.15: Definition of the damper characteristics

For a given axle, the damper characteristics are the same for both sides of the vehicle. The damper is modeled as a component that generates a force when being compressed / deflected (reaction to the change of velocity). Its length variation in function of the wheel travel is defined in the kinematics (see [section 5.11 'Suspension: Kinematics' on page 166](#)).

For additional information, see the Reference Manual, section "Dampers".

**Mode** Either the damper characteristic is specified by a single coefficient (mode *Characteristic Value*) or by a table (mode *Look-Up Table 1D*).

**Damping (N\*s/m)** For each front and rear axle, the damper characteristic is defined either by a coefficient or by a table of values according to the selected mode.

The characteristics of the damper are split into two domains: push and pull. For each domain you can define different damping. The push domain corresponds to a positive damper speed, which means that the damper is being compressed.

**Amplification (-)** This parameter enables to scale the damper characteristics very quickly instead of modifying the whole table. You can also use this parameter to convert the values entered to the units required by CarMaker.

## 5.9 Suspension: Buffer

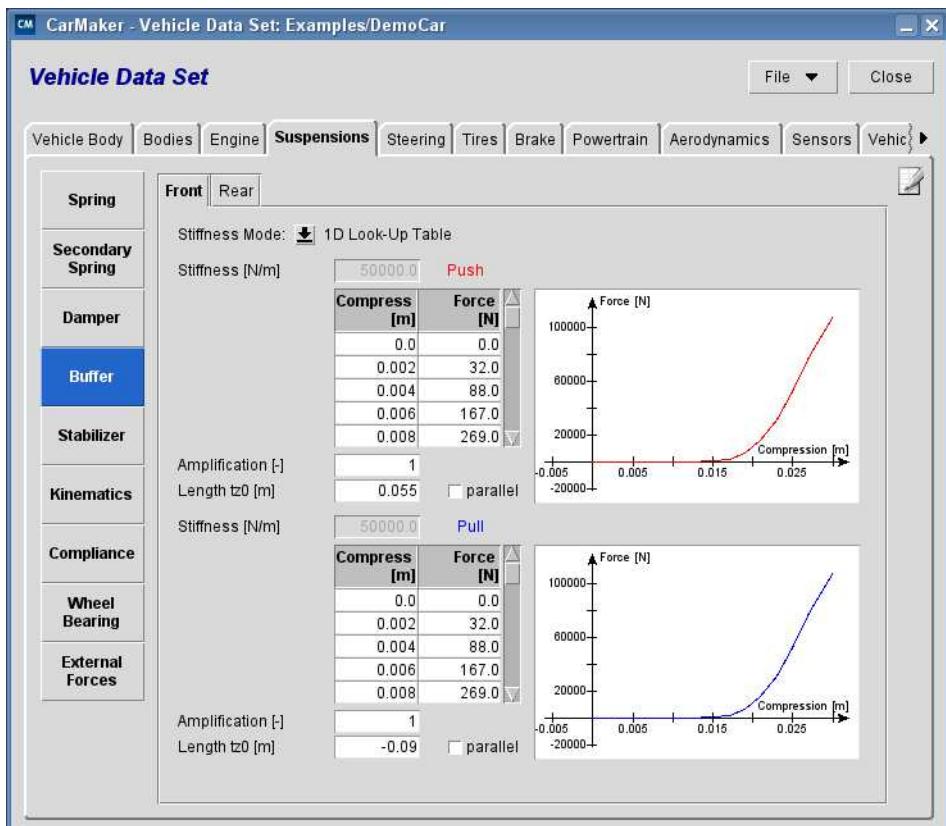


Figure 5.16: Definition of the buffer characteristics

For a given axle, the buffer characteristics are the same for both sides of the vehicle. The buffer is modeled as a spring which is activated only at a defined wheel travel. Its length variation when activated is defined in the kinematics (see [section 5.11 'Suspension: Kinematics' on page 166](#)).

For additional information see the section *Buffers* in the Reference Manual.

The buffers are used to limit the wheel travel in one direction or in both up and down. If the wheel travelled far enough to hit one of the buffers (for instance the push buffer), the buffer acts like an additional spring.

That is why you have to parameterize two buffers per axle (thus 4 in total), a stiffness for each buffer, and the wheel travel from which the buffers are activated for both positive and negative direction.



Note that you may parameterize the buffer influence directly by the spring stiffness. If you have already added the buffer stiffness to the spring stiffness, you do not need the buffer model any longer. In that case you just need to deactivate the real buffer model by setting the amplification factor to "0" (see below).

- Mode** The buffer stiffness can be parameterized by a coefficient (mode *Characteristic Value*) or by a table of values (mode *Look-Up Table 1D*).

**Buffer Stiffness (N/m)** For each front and rear axle, the buffer stiffness is defined either by a coefficient or by a table of values according to the selected mode.

The characteristics of the buffer are split into two domains: push and pull, where the push domain corresponds to a positive wheel travel.

#### Amplification (-)

This parameter enables to scale the buffer stiffness very quickly instead of modifying the whole table. You can also use this parameter to convert the values entered to the units required by CarMaker.

#### Length tz0 (m)

This length is the wheel travel in the vertical direction from which the buffer is activated (z axis of the frame  $Fr1$ ), calculated from the position in the design configuration (see [Figure 5.17: Buffer elements at the left wheel](#)).

#### tz0 parallel

If this flag is activated, the compression  $tz0$  is for parallel compression. Without this option,  $tz0$  is calculated by varying only one wheel travel whereas the opposite wheel is kept uncompressed..

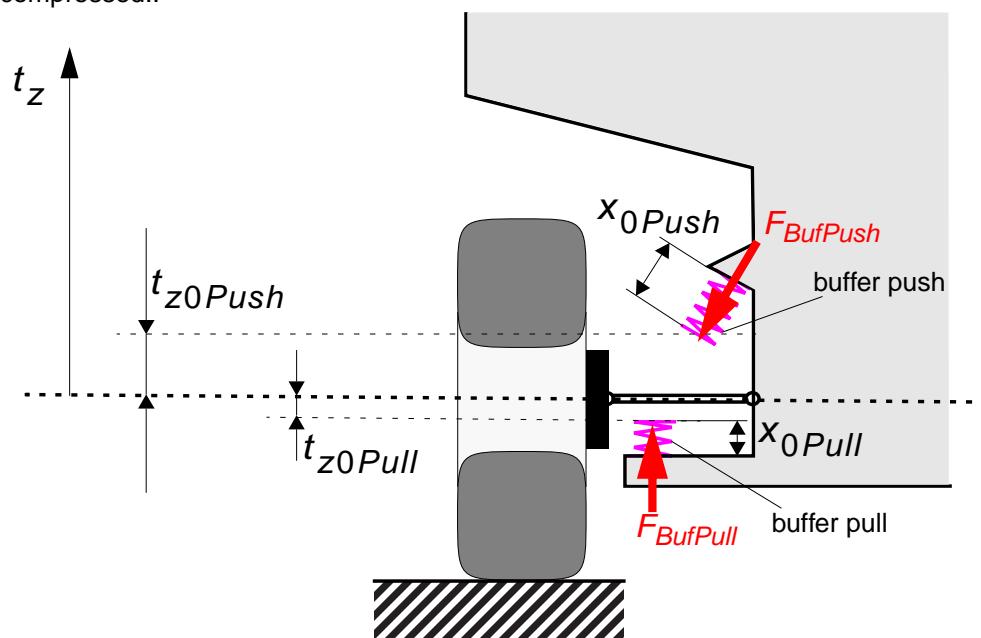


Figure 5.17: Buffer elements at the left wheel

## 5.10 Suspension: Stabilizer

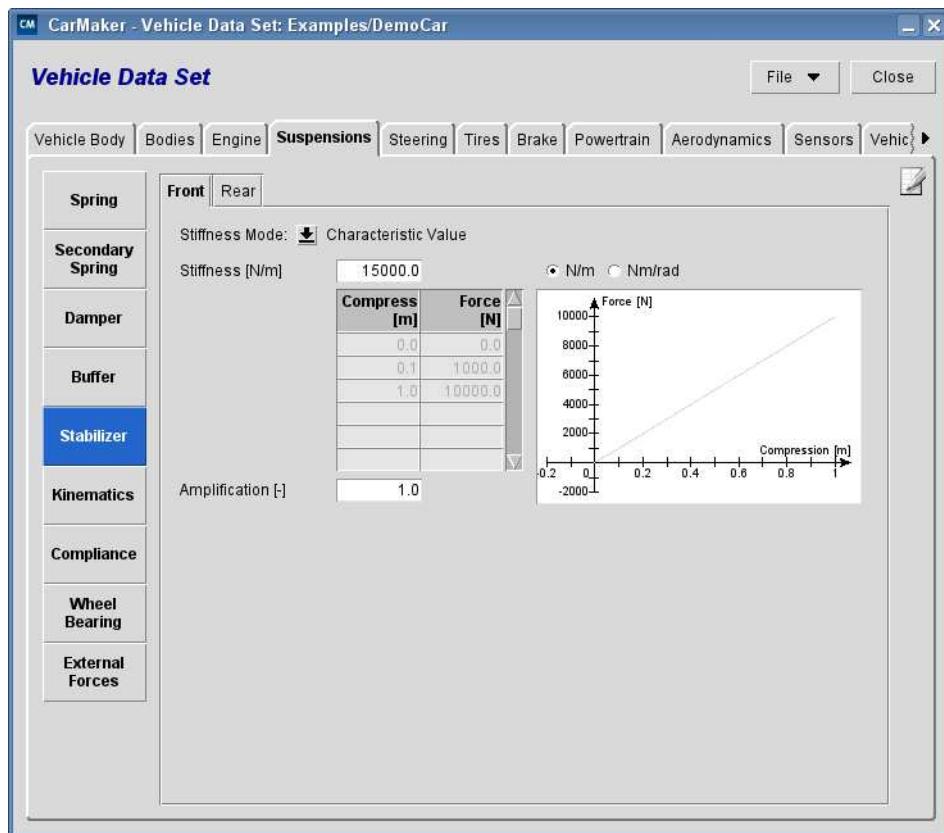


Figure 5.18: Definition of the stabilizer characteristics

At this point, only the properties of the stabilizer element are specified. Its angle or length variation is defined in the kinematics (see [section 5.11 'Suspension: Kinematics' on page 166](#)).

For additional information, see the Reference Manual, section "Suspension Roll Stabilizer/Anti-Roll Bar".

**Mode** For each front and rear axle, the stiffness of the stabilizer can either be defined with a simple coefficient (mode *Characteristic Value*) if the stabilizer force is a linear function, or in a table (mode *Look-Up Table 1D*) otherwise.

**Stiffness  
(N/m)  
or (Nm/deg)** This is the stiffness of the stabilizer, it either is a simple coefficient or a table according to the selected parameter *Mode*. Two options are available:

- the stabilizer as a virtual spring: unit of the stiffness is N/m.  
In that case, the displacement typically considered is the movement of the stabilizer end rod. This displacement should be very small.  
Attention: the stiffness depends on the place where the force is applied. Please find further information in the Reference Manual.
- the stabilizer is a real torsional component: the unit of the stiffness is Nm/rad.

**Amplification  
(-)** This parameter enables to scale the stiffness of the stabilizer very quickly instead of modifying the parameter itself. You can also use this parameter to convert the values entered to the units required by CarMaker.

## 5.11 Suspension: Kinematics

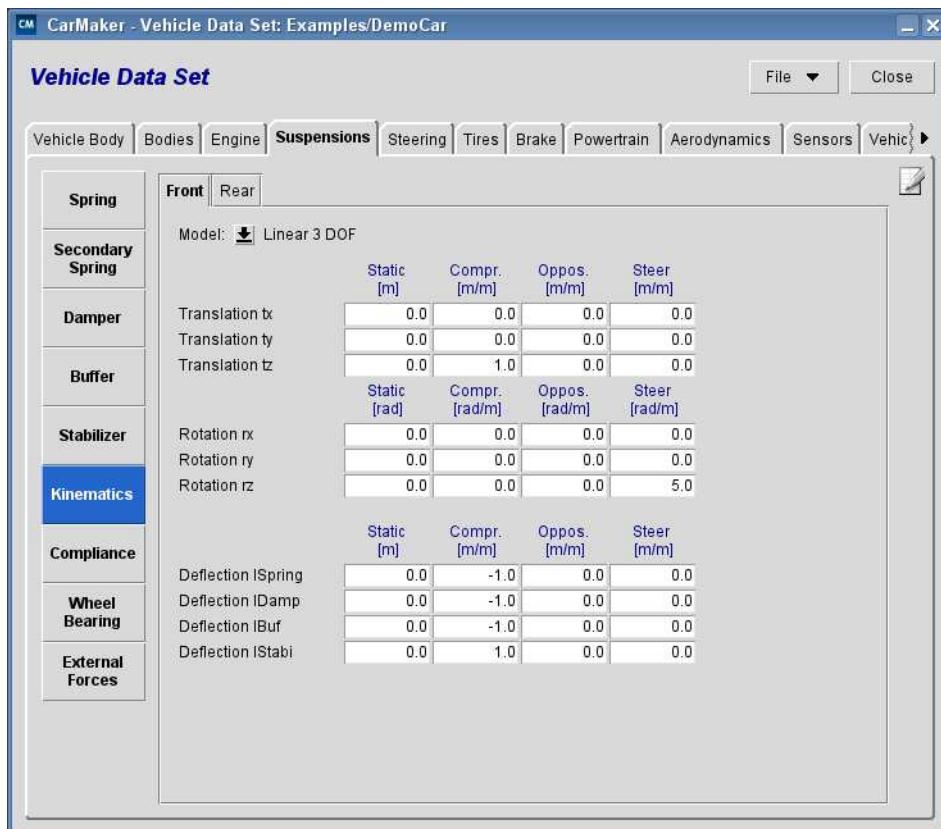


Figure 5.19: Definition of the kinematics

### 5.11.1 Basic Concept

Basically, CarMaker ignores the current geometry of the axle. What CarMaker does know, is the original position of the wheel (parameterized in the tabs *Bodies* and *Vehicle Body*) and the variations of these positions according to the wheel travel and the steering rack displacement. The position variations are described precisely by the kinematics model. For several states of wheel compression (and steering rack positions in case of a steered axle) the resulting position of the wheel center is defined by its translation in x, y, z and rotation around the x, y, z axes in relation to the static state.

In addition to the calculation of the wheel position, the kinematics model also includes the length variation of the spring, damper, buffer and ARB against the same variables (wheel travel and steering rack movement).

Knowing these kinematics tables, CarMaker can calculate the forces applied to the wheel center, for any independent axle or even for a rigid rear axle (see the description of the *Model*).

For additional information about the kinematics model, see the Reference Manual, section "Kinematics - Overview" and particularly section "Kinematics Models". For additional information about the internal degree of freedom, read the Reference Manual, section "Describing Kinematics with Generalized Coordinates".

For additional information about the calculation of the static equilibrium state and the start values used for the kinematics tables, please read the [section 'Calculation and Configuration of the Static Equilibrium State' on page 177](#).

## 5.11.2 Kinematic Models Overview

Various kinematics descriptions are available. The kinematics have to be specified for each the front and rear axle separately. Usually, a suspension is symmetrical, but a non-symmetrical axle can also be parameterized using the mode *External File*.

- Model** In general, two categories of kinematics characteristics are available which can be selected under *Model*. There are different linear kinematics descriptions with one, two or three degrees of freedom plus the *External SKC File* and *External MBS File* model for non-linear characteristics:

Table 5.6: Kinematics Models

Mode	Description	Axle
Linear 1 DOF	Linear uni-dimensional kinematics. The kinematics only depend on the wheel travel. See section 5.11.3 'Linear Kinematics Models'.	rear
Linear 2 DOF	Linear two-dimensional kinematics. The kinematics depend on the wheel travel and the steering rack displacement. This mode can be used to parameterize a steered front axle for an independent wheel suspension. See section 5.11.3 'Linear Kinematics Models'.	front
	Linear two-dimensional kinematics. The kinematics depend on the wheel travel and the movement of the opposite wheel. This mode can be used to parameterize a rigid or semi rigid rear axle. See section 5.11.3 'Linear Kinematics Models'.	rear
Linear 3 DOF	Linear three-dimensional kinematics. The kinematics depend on the wheel travel, the steering rack displacement and the movement of the opposite wheel. This mode can be used to parameterize a steered rigid or semi rigid front axle. See section 5.11.3 'Linear Kinematics Models'.	front
External SKC File	Non-linear kinematics. The kinematics are parameterized by the kinematics tables written to an external file.  In this case the values of each variable are defined for numerous steps of the wheel travel (e.g. from max. to min. bouncing, at step size 5mm) and optionally by the steering rack displacement or the wheel travel of the opposite wheel. See section 5.11.4 'Non-Linear Kinematics Model'	front or rear
External MBS File	Non-linear kinematics. The kinematics are modeled by a multi-body system. See section 5.11.5 'Multi Body Suspensions Overview'	front or rear

- The various *Linear X DOF* models are defined by two values (see [Table 5.7: Parameters of linear kinematic model](#)):
  - *Static*: the wheel center position when the vehicle is standing on the road (e.g. some suspensions use static camber or toe-in)
  - *Compr.*: a coefficient defining the gradient of the wheel movement over wheel travel
  - *Steer*: a coefficient defining the gradient of the wheel movement over steering rack travel
  - *Oppos*: a coefficient defining the gradient of the wheel movement over wheel displacement of the opposite wheel
- The *Linear 2 DOF* model can be used for both front and rear axles:

- if it is chosen for a front axle, the kinematics will depend on the wheel travel and the steering rack displacement.
- if it is chosen for a rear axle, the kinematics will depend on the wheel travel and the movement of the opposite wheel (rigid or semi-rigid axle).
- The *Linear 1 DOF* model must only be used for a rear axle: the kinematics only depend on the wheel travel.
- The *Linear 3 DOF* model can only be used for a steered axle.
- The non-linear model *External SKC File* can be used for both rear and front axles. With this mode, you parameterize exactly the same variables as with the other models (tx, ty,... see [Table 5.9: Parameters of non-linear kinematic model](#)). But in this case no single ratio is defined between the variation of this parameter (e.g. camber angle rx) and the variation of the generalized coordinates (e.g. wheel travel), but you describe the absolute values of this variable for numerous steps of the generalized coordinates (e.g. wheel travel).
- The non-linear model *External MBS File* can be used for both rear and front axles. It consists of a multi-body system model of the selected suspension type (e.g. McPherson axle) and its parameterization. In contrast to the other kinematics models available in CarMaker, this model kind can be parameterized without any previous simulation or measurement of the suspension kinematics.
- Additionally, there is the possibility to completely deactivate the kinematics. This is most commonly used if you want to implement your own kinematics description. Please find more information in the Reference Manual, section "SetZero".

### 5.11.3 Linear Kinematics Models

**tx, ty, tz (m)** Independently on the number of degree of freedoms (DOFs), the following 10 variables need to be parameterized as they are used by the model to define the wheel center movement. These parameters have a static value ( $c_{\text{static}}$  in [Figure 5.20](#)) and a coefficient describing the change according to the wheel travel and possibly according to the steering rack movement and/or according to the opposite wheel travel (depending on the DOF).

**rx, ry, rz (rad)**

**ISpring, IDamp,**

**IBuff, LStabi (m)**



In case of a linear kinematics description this coefficient is the gradient of the linear correlation (see also [Table 5.8: Coefficients for the linear kinematics model](#)). The following table gives an overview of the required parameters:

Table 5.7: Parameters of linear kinematic model

Parameter	Description	Unit (Static)	Unit (Compress)	Unit (Steering or Opposite)
Translation tx	wheel center variation	m	$\text{m}/\text{m}_{\text{wheel travel}}$	$\text{m}/\text{m}_{\text{steering rack travel}} \text{ or } \text{m}/\text{m}_{\text{opposite wheel travel}}$
Translation ty	track variation	m	$\text{m}/\text{m}_{\text{wheel travel}}$	$\text{m}/\text{m}_{\text{steering rack travel}} \text{ or } \text{m}/\text{m}_{\text{opposite wheel travel}}$
Translation tz	wheel travel variation	m	$\text{m}/\text{m}_{\text{wheel travel}}$	$\text{m}/\text{m}_{\text{steering rack travel}} \text{ or } \text{m}/\text{m}_{\text{opposite wheel travel}}$
Rotation rx	camber variation	rad	$\text{rad}/\text{m}_{\text{wheel travel}}$	$\text{rad}/\text{m}_{\text{steering rack travel}} \text{ or } \text{rad}/\text{m}_{\text{opposite wheel travel}}$
Rotation ry	spin angle variation	rad	$\text{rad}/\text{m}_{\text{wheel travel}}$	$\text{rad}/\text{m}_{\text{steering rack travel}} \text{ or } \text{rad}/\text{m}_{\text{opposite wheel travel}}$
Rotation rz	toe variation	rad	$\text{rad}/\text{m}_{\text{wheel travel}}$	$\text{rad}/\text{m}_{\text{steering rack travel}} \text{ or } \text{rad}/\text{m}_{\text{opposite wheel travel}}$

Table 5.7: Parameters of linear kinematic model

Parameter	Description	Unit (Static)	Unit (Compress)	Unit (Steering or Opposite)
Deflection ISpring	spring length variation	m	$m/m_{\text{wheel travel}}$	$m/m_{\text{steering rack travel}} \text{ or } m/m_{\text{opposite wheel travel}}$
Deflection IDamp	damper length variation	m	$m/m_{\text{wheel travel}}$	$m/m_{\text{steering rack travel}} \text{ or } m/m_{\text{opposite wheel travel}}$
Deflection IBuff	buffer length variation if activated	m	$m/m_{\text{wheel travel}}$	$m/m_{\text{steering rack travel}} \text{ or } m/m_{\text{opposite wheel travel}}$
Deflection IStabi	Stabilizer length (showing the vertical movement of the ARB at the point where the stiffness was measured) or angle variation, according to which unit you used for the ARB stiffness.	m or rad	$m/m_{\text{wheel travel}}$ or $\text{rad}/m_{\text{wheel travel}}$	$m/m_{\text{steering rack travel}} \text{ or } m/m_{\text{opposite wheel travel}}$ or $\text{rad}/m_{\text{steering rack dist}} \text{ or } \text{rad}/m_{\text{opposite wheel travel}}$

Let us now look in detail at the definition of the kinematic coefficients which have to be defined in the table for the linear kinematics modes:

Table 5.8: Coefficients for the linear kinematics model

Column	Description
Static	The values in this column indicate the static design parameters, e.g. when the wheel travel is zero.  The values for tx, ty, and tz should remain zero here, as otherwise this would lead to a change in the position of the wheels and wheel carriers. Sometimes, a value different from zero can be defined, e.g. to check a setup variant quickly. For rx, ry, and rz the values written here define the static camber, caster and toe in/out.
Compress	The values in this column give the variation of a parameter (e.g. the camber angle rx, in rad), per meter of wheel travel variation.
Steering	The values in this column give the variation of a parameter (e.g. the camber angle rx in rad), per meter of steering rack displacement. This column is used only for a front axle (mode <i>Linear 2 DOF</i> ).
Opposite	The values in this column give the variation of a parameter (e.g. the camber angle rx in rad), per meter of wheel travel of the opposite wheel. This column is used only for a rigid axle (mode <i>Linear 2 DOF</i> for a rear axle or <i>Linear 3 DOF</i> for a front axle).

Besides the translation of the wheel center, the change in length of spring, damper, buffer and ARB has to be parameterized. As you have seen, you have to parameterize the behavior of the ARB. As explained in [section 5.10 'Suspension: Stabilizer' on page 165](#), you can define the stiffness of the ARB in two different units:

- Stiffness in N/m: in this case you have to parameterize IStabi in  $m/m_{\text{wheeltravel}}$
- Stiffness in Nm/rad: in this case you have to parameterize IStabi in  $\text{rad}/m_{\text{wheeltravel}}$





Once you have parameterized the kinematics, we suggest you to check the corresponding kinematics diagrams in the Model Check (see the [section 6.4 'Model Check' on page 360](#)).

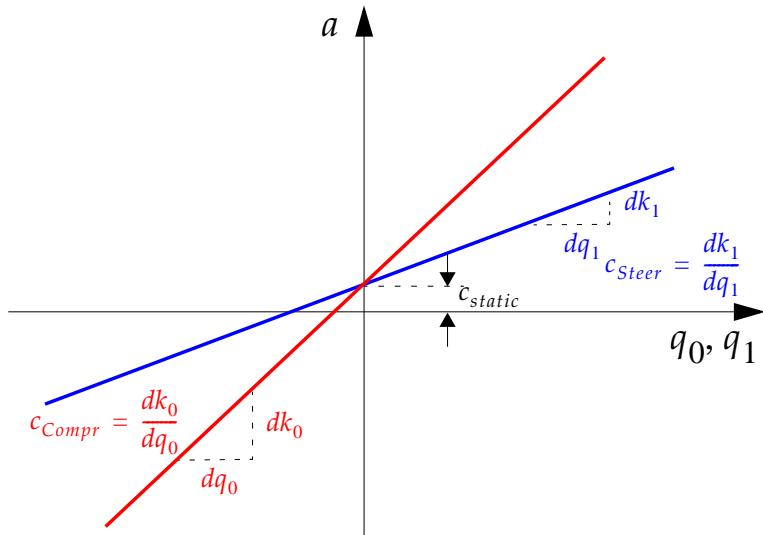


Figure 5.20: Definition of kinematics coefficients for the linear kinematics models

#### 5.11.4 Non-Linear Kinematics Model

In case of a non-linear kinematics description an external file needs to be loaded. To activate this model and to select the so called "skc file", select *External SKC File*.

This model basically requires the same input as the *Linear* models: The translations and rotations of the wheel center point in dependence on the degree of freedom (wheel travel, steering rack movement or wheel travel of opposite wheel). However, the movement of the wheel center is not described as linear function but by giving measurement points for discrete states of the DOF.



Before trying to parameterize the *External SKC File* model, we recommend to read and understand how the *Linear X DOF* mode works.

Using the non-linear kinematics model, you have to parameterize the kinematic characteristics for each of the 10 kinematics variables, and not only one or possibly two coefficients as described in the previous paragraph.

For a start we suggest you to look at the front axle example *McPherson\_FrontAxle.skc*. For this, load the "DemoCar" vehicle data set, open the vehicle data set editor and display the kinematic window. Then look at the front axle: the mode *External SKC File* is already activated. Click on the button *Select* next to the field *K&C File*. Keep the selection of the *McPherson\_front.skc* file and then click on *Edit*.

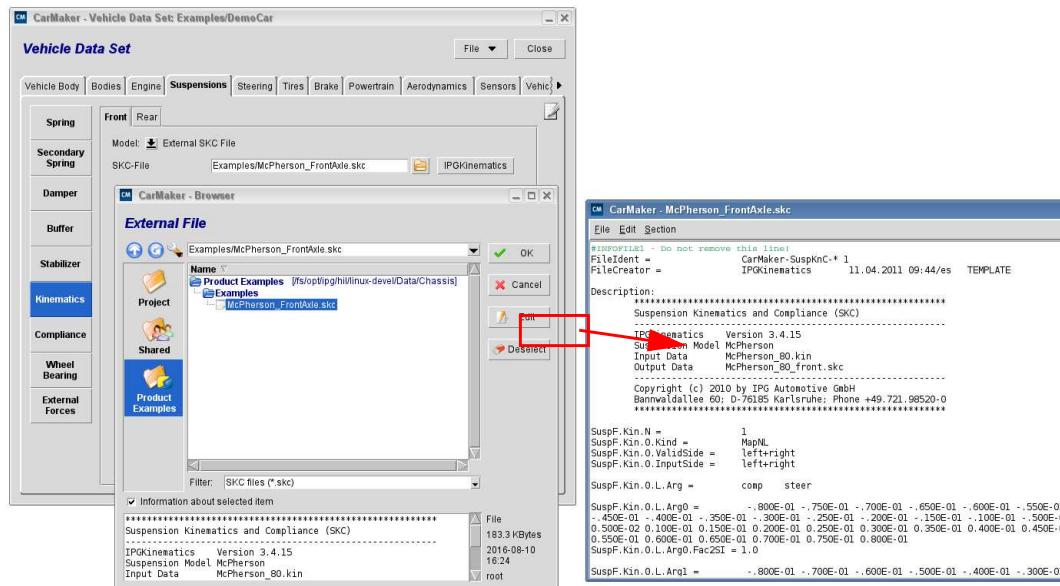


Figure 5.21: Opening the skc file from the vehicle data set editor

The skc file contains a huge matrix with measurement points, that describe the wheel center movement for different measurement states:

Table 5.9: Parameters of non-linear kinematic model

Parameter	Unit (Static)	Description
%i0	m	first degree of freedom: wheel travel (wheel pad movement)
%i1	m	second degree of freedom: steering rack movement or wheel travel of opposite wheel (depends on the model)
tx	m	translation of wheel center in x direction
ty	m	translation of wheel center in y direction
tz	m	translation of wheel center in z direction
rx	rad	rotation angle of wheel center around x axis
ry	rad	rotation angle of wheel center around y axis
rz	rad	rotation angle of wheel center around z axis
ISpring	m	absolute spring length
IDamp	m	absolute damper length
IBuff	m	absolute buffer length
IStabi	m or rad	absolute stabilizer length (showing the vertical movement of the ARB at the point where the stiffness was measured) or deflection angle, depends on the unit you used for the ARB stiffness.

Parameterizing non-linear kinematics requires a power user level. For this, you should understand the kinematics model very well and in detail by reading the Reference Manual, section "Kinematics Models".

If the explanations of this documentation are not sufficient, contact the CarMaker Service Team. You can also contact your sales partner at IPG Automotive to let you know about a training on this topic.



Once you have parameterized the kinematics, we suggest you to check the corresponding kinematics diagrams in the Model Check (see the [section 6.4 'Model Check' on page 360](#)).

## 5.11.5 Multi Body Suspensions Overview

The model *External MBS File* is based on a high resolution multi-body system (MBS) suspension for the front and rear axles. With this particular functionality, the user does not have to use an external MBS software such as IPGKinematics, or make use of a K&C measurement to arrive at a suitable file format for simulation. The MBS Suspension model consists of defining the hard-points, masses, inertias, etc. of the elements that make up the suspension system. The MBS suspension model then behaves along the lines of a virtual suspension test rig, thereby generating all the Kinematics and Compliance information that required, on its own. The interface for the MBS suspension is, however, the same as that used for the SKC method of Kinematics and Compliance mapping as explained in the section *Suspension Kinematics and Compliance* in the Reference Manual.

Please select the Model External MBS File model from the drop-down menu on the Kinematics tab of the Vehicle Data Set editor. If an MBS file has already been parameterized and you have an MBS-File available, please use the file browser and choose the appropriate file.

### General

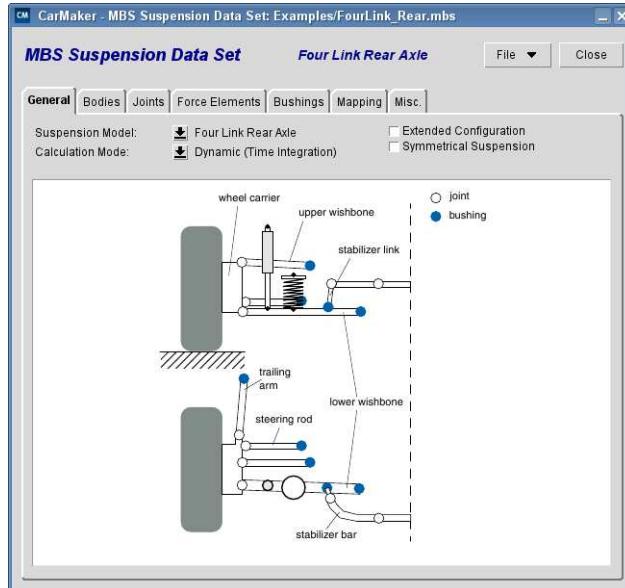


Figure 5.22: Definition of General MBS parameters

#### Suspension Model

The type of suspension model template that needs to be used can be chosen from the ones that are available here. The two available options are the "McPherson Front Axle" and the "FourLink Rear Axle". In case you require another type of suspension template, please get in touch with the CarMaker Service Team.

**Calculation Mode** The "Dynamic (Time integration)" mode consists of a time integration of the MBS suspension during the CarMaker simulation, which leads to a higher resolution suspension model that includes all the dynamic effects due to the instantaneous forces and torques acting on the suspension to calculate the generalized coordinates.

The "Static (2D Look-Up Table)" mode is similar to the External SKC File method of Kinematics and Compliance, but does not require any other external MBS software or K&C Measurement to generate an SKC file. The MBS suspension model automatically calculates a static SKC map during the preparation phase, based on the way you parameterize the MBS suspension and then uses this static map during the entire simulation.

**Symmetric Suspension** When this option is chosen, then only the left side of the MBS suspension model has to be parameterized and the model then automatically mirrors the parameterization for the right side as well.

## Bodies

The Masses and the Inertias of the individual elements of the suspension model should be provided here.



Please be aware that these masses are not automatically added to the global wheel carrier mass and the vehicle body mass in the "Vehicle Data Set". This has to be done by the user. The mass parameterization in the MBS suspension will, however, as shown in the table below, be divided into their individual masses and inertias.

**Mass (kg) Ixx (kgm<sup>2</sup>) Iyy (kgm<sup>2</sup>) Iyy (kgm<sup>2</sup>)** The masses of the individual suspension elements must be provided in the Mass boxes. Under *Ixx*, *Iyy* and *Izz* the respective moments of inertia of these body masses can be defined in the axis orientation of frame Fr1.

**Graphic** This button provides a schematic representation of the suspension type that has been chosen, to enable a better understanding of the hard-points, elements, and their configuration while parameterizing the MBS suspension.

	Mass [kg]	Ixx [kgm <sup>2</sup> ]	Iyy [kgm <sup>2</sup> ]	Izz [kgm <sup>2</sup> ]	Graphic
Wheel carrier left	10.0	0.03	0.06	0.04	
Wheel carrier right	10.0	0.03	0.06	0.04	
Wishbone lower left	5.4	0.2	0.1	0.1	
Wishbone lower right	5.4	0.2	0.1	0.1	
Wishbone upper left	1.5	0.02	0.001	0.02	
Wishbone upper right	1.5	0.02	0.001	0.02	
Spring left	2.0	0.01	0.01	0.003	
Spring right	2.0	0.01	0.01	0.003	
Damper left	2.5	0.05	0.05	0.004	
Damper right	2.5	0.05	0.05	0.004	
Trailing arm left	2.5	0.004	0.05	0.05	
Trailing arm right	2.5	0.004	0.05	0.05	
Steering rod left	1.0	0.01	2e-4	0.01	
Steering rod right	1.0	0.01	2e-4	0.01	
Stabilizer bar	4.0	0.4	0.014	0.4	
Stabilizer link left	0.1	0.004	1e-5	0.004	
Stabilizer link right	0.1	0.004	1e-5	0.004	

Figure 5.23: Definition of the masses and inertias of components in the MBS suspension

If the explanations of this documentation are not sufficient, please contact the CarMaker Service Team. You can also contact your sales partner at IPG to let you know about a training on this topic.



Once you have parameterized the kinematics, we suggest you to check the corresponding kinematics diagrams in the Model Check (see the [section 6.4 'Model Check' on page 360](#)).

## Joints

The coordinates of all the hard-points of the suspension elements (apart from the Spring and Damper) can be parameterized here in terms of x, y and z-coordinates of the  $Fr_{susp}$ . Please take note that the origin of the  $Fr_{susp}$  is located at the centre of the axis that joins the two wheel centers of the left and right wheel. For example, the  $Fr_{susp}$  for the front axle would therefore be the centre of the axis that joins the front left and front right wheel centers with respect to  $FrD$ .

### Origin of the vehicle frame $Fr1$

The relative coordinates of the frame  $Fr1$  with respect to the  $Fr_{susp}$   
(e.g: x = -3.35m, y = 0.0m, z = -0.3m)

### x (m) y (m) z (m)

The joints that are provided depend completely on the type of suspension that has been chosen. Please refer to the section *Suspension Kinematics and Compliance* in the Reference Manual for more information regarding the specific joints for each type of suspension. The positions of the joints that comprise the particular suspension setup chosen by you can be parameterized here in the form of x, y and z-coordinates with respect to the frame  $Fr_{susp}$ .

### Graphic

This button provides a schematic representation of the suspension type that has been chosen, to enable a better understanding of the hard-points, elements, and their configuration while parameterizing the MBS suspension.

## Force Elements

The coordinates of the hard-points of the Force Elements, i.e., the Spring and Damper elements can be provided here in terms of x, y and z-coordinates of the  $Fr_{susp}$ . Please take note that the origin of the  $Fr_{susp}$  is located at the centre of the axis that joins the two wheel centers of the left and right wheel. For example, the  $Fr_{susp}$  for the front axle would therefore be the centre of the axis that joins the front left and front right wheel centers with respect to  $FrD$ .

### x (m) y (m) z (m)

The hard-points of the Force Elements that are provided depend completely on the type of suspension that has been chosen. The positions of the joints that comprise the particular suspension setup chosen by you can be parameterized here in the form of x, y and z-coordinates with respect to the frame  $Fr_{susp}$ . Please refer to the Reference Manual, section "Suspension Kinematics and Compliance" for more information regarding the specific Force Element hard-points for each type of suspension as well as their corresponding further options for parameterization.

### Graphic

This button provides a schematic representation of the suspension type that has been chosen, to enable a better understanding of the hard-points, elements, and their configuration while parameterizing the MBS suspension.

## Bushings

The parameters for all bushings of the wheel suspension are entered in the "Bushings" tab. The various bushings that are a part of the suspension type that has been chosen are shown in a selection pane of the left. The corresponding parameterization for that particular bushing can then be done based on the various options provided.

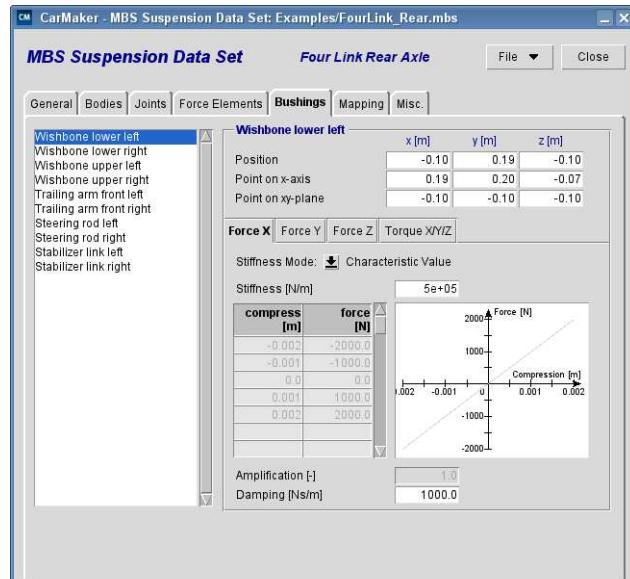


Figure 5.24: Definition of the bushing parameters in the MBS suspension

## Bushing Axis System

A "Bushing Axis System" is calculated from the "Bushing Center Point", the "Point on Bearing Axle" and the "Angle at Bushing Axle" (see [Figure 5.25 on page 175](#)). The bushing axle of the bushing is always the X-axis of the bushing axis system. It is favorable when the X-axis of the bushing axis system is oriented in the positive direction of the X-axis of the fixed vehicle axis system, irrelevant of the installation position of the bushing in the vehicle.

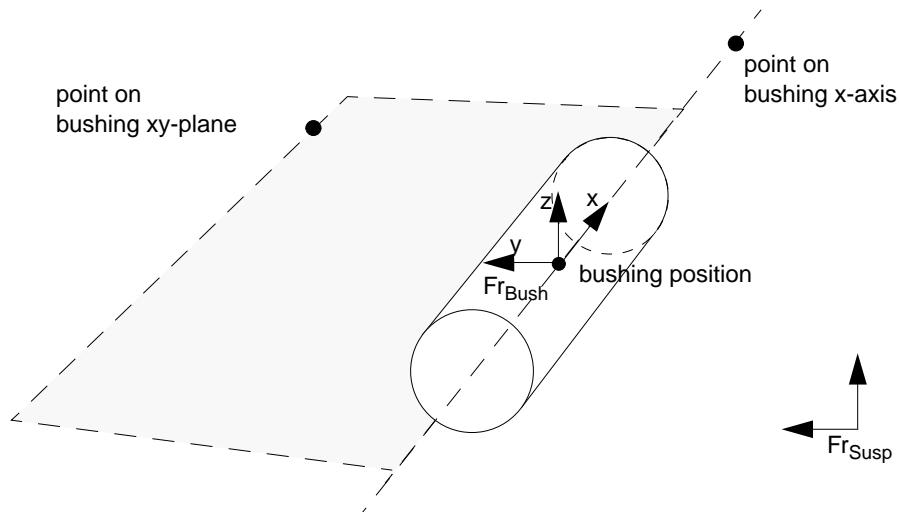


Figure 5.25: Bushing position and orientation

<b>Position</b>	Specifies the position of the bushing center expressed in $F_{r,susp}$ coordinates (please refer to <a href="#">Figure 5.25</a> ).
<b>Point on x-axis</b>	Specifies the position of a point located on the bushing x-axis expressed in $F_{r,susp}$ coordinates (please refer to <a href="#">Figure 5.25</a> ).
<b>Point on xy-plane</b>	Specifies the position of a point locating on the bushing xy-plane expressed in $F_{r,susp}$ coordinates (please refer to <a href="#">Figure 5.25</a> ).
<b>Force X</b>	The bushing spring characteristic "Stiffness [N/m]" can be defined with a Characteristic Value or with a 1D-lookup table (can be chosen from the drop-down menu of "Stiffness Mode") describing the non-linear spring behavior. This characteristic translates compression along the bushing x-axis to spring force.  The bushing damping characteristic "Damping [Ns/m]" can be defined as a constant value. This characteristic translates compression velocity around the bushing x-axis to spring force.
<b>Force Y</b>	The bushing spring characteristic "Stiffness [N/m]" can be defined with a Characteristic Value or with a 1D-lookup table (can be chosen from the drop-down menu of "Stiffness Mode") describing the non-linear spring behavior. This characteristic translates compression along the bushing y-axis to spring force.  The bushing damping characteristic "Damping [Ns/m]" can be defined as a constant value. This characteristic translates compression velocity around the bushing y-axis to spring force.
<b>Force Z</b>	The bushing spring characteristic "Stiffness [N/m]" can be defined with a Characteristic Value or with a 1D-lookup table (can be chosen from the drop-down menu of "Stiffness Mode") describing the non-linear spring behavior. This characteristic translates compression along the bushing z-axis to spring force.  The bushing damping characteristic "Damping [Ns/m]" can be defined as a constant value. This characteristic translates compression velocity around the bushing z-axis to spring force.
<b>Torque X/Y/Z</b>	The bushing characteristic "Rot. Stiffness [Nm/rad]" specifies the three bushing rotational spring characteristic values around the bushing x-, y- and z-axis. This characteristic translates rotational compression around the bushing axis to spring torque.  The bushing characteristic "Rot. Damping [Nms/rad]" specifies the three bushing rotational damping characteristic values around the bushing x-, y- and z-axis. This characteristic translates rotational compression velocity along the bushing axis to spring torque.  The bushing characteristic "Preload Torque" specifies the bushing preload torque around the bushing x-, y- and z-axis.

## Misc.

<b>Static Camber Angle (deg)</b>	The static camber angle for left and right side can be provided here.
<b>Static Toe Angle (deg)</b>	The static toe angle for left and right side can be provided here.
<b>Description</b>	You can provide a brief description of the model that you have parameterized. This description is then shown on the file browser and helps you in choosing the right MBS-File for a particular simulation.

**Additional Parameters** If you have developed a special model, you may need to parameterize it. In this case, you can write some additional parameters in this field that will be written in the MBS-File, and then read the additional parameters directly from the MBS-File.

## 5.11.6 Additional information

Let us now give some additional information useful to understand how to parameterize the kinematics model:

- To parameterize the kinematics, consider a left wheel and a right hand orthogonal axis system.
- Signs: please read the Reference Manual, section "CarMaker Axis Systems" to get more information about the coordinate system  $F_2$  used to parameterized the kinematics. According to this convention:
  - a toe in has a positive sign,
  - a spin angle oriented from lower front to upper rear has a negative sign,
  - a camber angle with the upper part of the wheel oriented inward has a negative sign.
- $q_0$ , referred to as *wheel travel*, is the generalized coordinate describing the wheel movement under bouncing. The orthogonal movement of the wheel  $t_z$  (the z-coordinate in  $F_1$ ) is not the same as  $q_0$  in an accurate kinematics model. E.g. a steering rack movement might also lead to a vertical translation of the wheel center point.
- How to get the required kinematics data? You can either take measurements on a K&C rig, or calculate it with a MBS software (e.g. IPGKinematics). For additional information, see the Reference Manual, section "Brief Introduction to the Measurement Procedure of K&C parameters.

## Calculation and Configuration of the Static Equilibrium State

The kinematics tables directly influence the parameterization of the body mass positions and the spring's free length.

To parameterize the static state, there are two possible approaches:

- The vehicle is parameterized in the static equilibrium state (e.g. position of the masses), the forces applied by the springs in this state are also parameterized, and the model calculates the corresponding free length based on the kinematics table. This approach has the drawback that the forces have to be redefined each time the vehicle weight is changed.
- The vehicle is parameterized for a given design state (which can be the same for all vehicle variants). The current free length of the spring is also parameterized, and, thanks to the kinematics tables, the software automatically calculates the static equilibrium state. This solution is used in CarMaker since you have to get the spring and kinematics properties (stiffness, free length, spring length variations) only once, and CarMaker will always calculate the corresponding static equilibrium state.

The first approach is clearly less convenient and that is why the second approach was implemented in CarMaker.

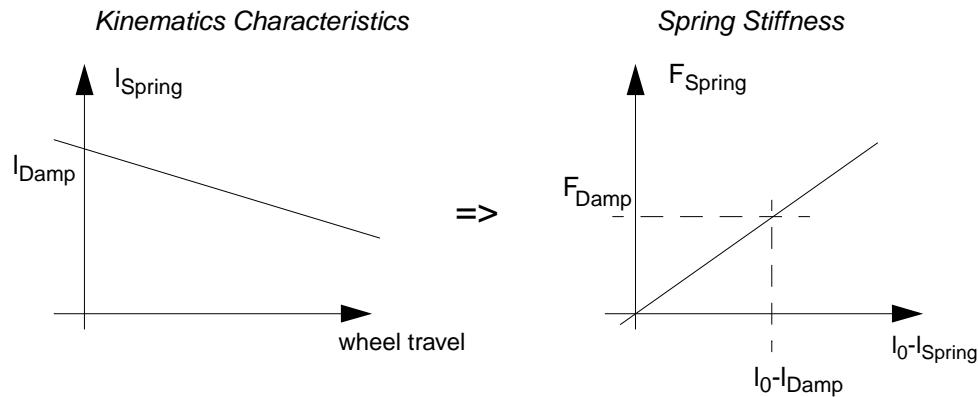
Before going further, we have to define what the design state actually is. The design state is a state ahead of the static equilibrium state. The wheels have a given position compared to the vehicle body which does not respect the static force equilibrium. It is a virtual state used only for design purposes (independently from CarMaker). Note that you can design your vehicle to match exactly the static equilibrium state. However, this position is the static equilibrium state only for a given vehicle, and if its weight changes, then this position is only one design state among others.

To sum up, CarMaker will automatically calculate the static equilibrium state ahead of every simulation (during the preparation phase). For example, according to the raw kinematic and spring parameterization the software will find the position for which there is an equilibrium of the forces.

Let us have a look at the next two examples. As a matter of fact, we strongly recommend to parameterize the vehicle and its kinematics according to the first example.

### First Example

Let us consider the following starting condition and its effect on the calculation of the static equilibrium state:



At the beginning of the preparation phase:

$wheel\ travel = 0 \Rightarrow l_{Spring} = l_{Damp} \Rightarrow (l_0 - l_{Spring}) = (l_0 - l_{Damp}) \Rightarrow F_{Spring} = F_{Damp} \Rightarrow (\Sigma F_{vehicle} \neq 0) \Rightarrow$  Application of small movements  $\Rightarrow$  wheel travel is increased  $\Rightarrow l_{Spring} = \dots\dots$  and this up to the position where  $\Sigma F_{vehicle} = 0$

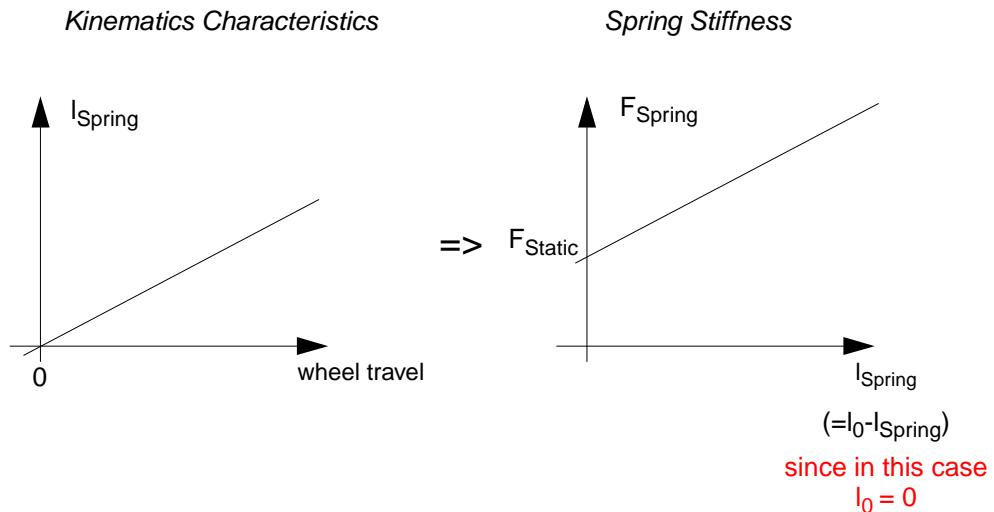
In this case, all parameters are given in a default design position of the axle. In respect to the vehicle weight, the correct static equilibrium state will be found by CarMaker. Of course, now the position of the vehicle mass (in particular the height) has to be given according to the design state you chose.

If you already have all masses' positions and kinematics data in the static equilibrium state, it is no problem to use the approach nonetheless: just regard this state as a special design state. All what you have to do is to shift the characteristics of  $l_{Spring}$  down, so that the value of  $l_{Spring}$  at wheel travel = 0 corresponds to the spring length in the static equilibrium state, which generates a  $F_{Spring}$  that already corresponds to the force applied by the spring in the equilibrium state. For this there is a parameter in the model *External SKC File (\*.Data.Offset*, see the Reference Manual, section "MapNL" for the parameterization of the model *External File/MapNL*).

### Second Example

Even though you can parameterize the masses' positions in the static equilibrium state with the first solution (first example), there is an alternative explained in this approach.

Let us consider that the parameter  $l_0$  is not the free length any longer. There are two possibilities that can be considered here to parameterize the kinematics with regard to the  $l_{Spring}$  and  $l_0$ . The first is to not provide the  $l_{Spring}$  values in the *External SKC File* and untick the option  $l_0$ . This way CarMaker internally calculates the corresponding  $l_{Spring}$  and  $l_0$  values based on parameter variation using the Regula-Falsi method. The second possibility is to provide the  $l_{Spring}$  values as the absolute length measured during the vehicle Kinematics and Compliance testing and let CarMaker internally calculate the  $l_0$  values again based on the Regula-Falsi method.



At the beginning of the preparation phase:

wheel travel = 0  $\Rightarrow l_{\text{Spring}} = 0 \Rightarrow (l_0 - l_{\text{Spring}}) = 0 \Rightarrow F_{\text{Spring}} = F_{\text{Static}} \Rightarrow \Sigma F_{\text{Vehicle}} = 0$   
 $\Rightarrow$  Static Equilibrium State already reached.

This configuration should be used only to parameterize the masses' position in the static equilibrium state.

### 5.11.7 MBS: McPherson

This section explains the parameters specific to this MBS model. Please find more information about the general parameters in [section 5.11.5 'Multi Body Suspensions Overview' on page 172](#).

#### General

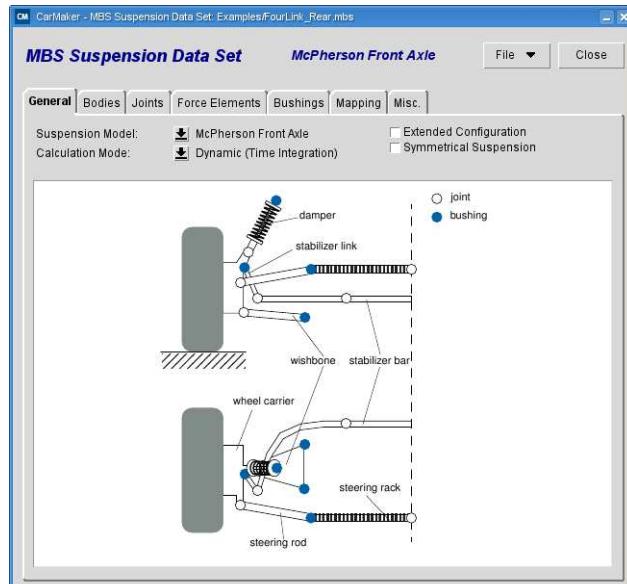


Figure 5.26: McPherson general parameters

**Extended Configuration** If checked Axle is mounted on a subframe.

## Bodies

Table 5.10: Definition of the individual suspension components for McPherson MBS

Name of the Body	Comments	Tab
Subframe	Measured mass of the Subframe if "Extended Configuration" was used.	Bodies
Wheel Carrier (x2) (left/right)	Measured mass of the Wheel Carrier element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies
Wishbone lower (x2) (left/right)	Measured mass of the lower Wishbone element of the suspension. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies
Damper (x2) (left/right)	Measured mass of the Damper element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies
Steering gearbox	Measured mass of the steering gearbox.	Bodies
Steering rack	Measured mass of the steering rack.	Bodies
Steering rod left (x2) (left/right)	Measured mass of the steering rod element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies
Steering bar	Measured mass of the steering bar.	Bodies
Steering link (x2) (left/right)	Measured mass of the steering link element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies

## Joints

Table 5.11: Definition of the individual suspension components for McPherson MBS

Name of the joint	Comments	Tab
Origin of the vehicle frame Fr1	The relative coordinates of the frame Fr1 with respect to the Fr <sub>susp</sub> (e.g: x = -3.35m, y = 0.0m, z = -0.3m)	Joints
Wheel carrier - Wishbone lower (x2) (left/right)	Hard point of wheel carrier to wishbone. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Joints
Wheel carrier - Steering rod (x2) (left/right)	Hard point of wheel carrier to steering rod. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Joints

Table 5.11: Definition of the individual suspension components for McPherson MBS

Name of the joint	Comments	Tab
Vehicle - Stabilizer bar (x2) (left/right)	Hard point of vehicle body to stabilizer. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Joints
Subframe - Stabilizer bar (x2) (left/right)	Hard point of subframe to stabilizer. Can be defined separately (if "Symmetric Suspension" has not been chosen and if "Extended Configuration" is checked)	Joints
Stabilizer bar - Stabilizer link (x2) (left/right)	Hard point of stabilizer bar to stabilizer link. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Joints

## Force Elements

Table 5.12: Definition of the individual suspension components for McPherson MBS

Name of the element	Comments	Tab
Spring force upper (x2) (left/right)	Hard point of upper spring force. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Force Elements
Spring force lower (x2) (left/right)	Hard point of lower spring force. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Force Elements
Damper force upper (x2) (left/right)	Hard point of upper damper force. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Force Elements
Damper force lower (x2) (left/right)	Hard point of lower damper force. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Force Elements

## Bushings

Table 5.13: Definition of the individual suspension components for McPherson MBS

Name of the bushing	Comments	Tab
Subframe front (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of subframe bushing front. Can be defined separately (if "Symmetric Suspension" has not been chosen and if "Extended Configuration" is checked)	Bushings
Subframe rear left (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of subframe bushing rear. Can be defined separately (if "Symmetric Suspension" has not been chosen and if "Extended Configuration" is checked)	Bushings
Steering gearbox (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of steering gearbox bushing. Can be defined separately (if "Symmetric Suspension" has not been chosen and if "Extended Configuration" is checked)	Bushings
Damper piston (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of damper piston. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Bushings
Wishbone lower front (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of lower wishbone bushing front. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Bushings
Wishbone lower rear (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of lower wishbone bushing rear. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Bushings
Steering rod (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of steering rod bushing. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Bushings
Steering link (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of steering link bushing. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Bushings

## Mapping

Table 5.14: Definition of the individual suspension components for McPherson MBS

Name of the element	Comments	Tab
Number of grid points	Number of grid points for the tableaus.	Mapping

Table 5.14: Definition of the individual suspension components for McPherson MBS

Name of the element	Comments	Tab
Maximum compression [m]	Maximum compression which will be used for the tableaus.	Mapping, Kinematics
Maximum steering rack travel [m]	Maximal steering rack travel which will be used for the tableaus.	Mapping, Kinematics
Maximum compression [m]	Maximum compression which will be used for the tableaus.	Mapping, Compliance
Maximum force [N]	Maximum force which will be used for the tableaus.	Mapping, Compliance
Maximum torque [Nm]	Maximum torque which will be used for the tableaus.	Mapping, Compliance

## Misc

Table 5.15: Definition of the individual suspension components for McPherson MBS.

Name of the element	Comments	Tab
Static Camber Angle [deg] (x2) (left/right)	Static camber angle. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Misc.
Static Toe Angle [min]	Static toe angle. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Misc.

## 5.11.8 MBS: Four Link

This section explains the parameters specific to this MBS model. Please find more information about the general parameters in [section 5.11.5 'Multi Body Suspensions Overview' on page 172](#).

## General

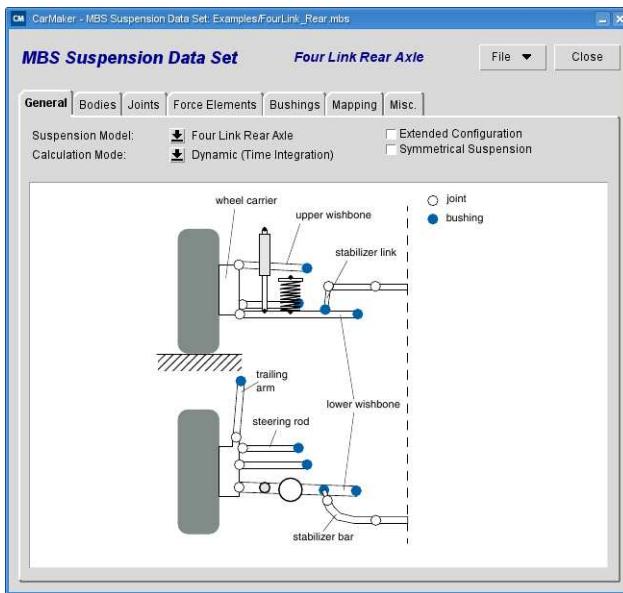


Figure 5.27: Four Link axle general parameters

**Extended Configuration** If checked Axle is mounted on a subframe.

## Bodies

Table 5.16: Definition of the individual suspension components for Four Link MBS

Name of the Body	Comments	Tab
Subframe	Measured mass of the Subframe if "Extended Configuration" was used.	Bodies
Wheel Carrier (x2) (left/right)	Measured mass of the Wheel Carrier element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies
Wishbone lower (x2) (left/right)	Measured mass of the lower Wishbone element of the suspension. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies
Wishbone upper (x2) (left/right)	Measured mass of the upper Wishbone element of the suspension. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies
Spring (x2) (left/right)	Measured mass of the Damper element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies
Damper (x2) (left/right)	Measured mass of the Damper element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies

Table 5.16: Definition of the individual suspension components for Four Link MBS

Name of the Body	Comments	Tab
Trailing arm (x2) (left/right)	Measured mass of the Trailing arm element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies
Steering rod left (x2) (left/right)	Measured mass of the steering rod element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies
Stabilizer bar	Measured mass of the stabilizer bar.	Bodies
Stabilizer link (x2) (left/right)	Measured mass of the stabilizer link element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies

## Joints

Table 5.17: Definition of the individual suspension components for Four Link MBS.

Name of the joint	Comments	Tab
Origin of the vehicle frame Fr1	The relative coordinates of the frame Fr1 with respect to the Fr <sub>susp</sub> (e.g: x = -3.35m, y = 0.0m, z = -0.3m)	Joints
Wheel carrier - Wishbone lower (x2) (left/right)	Hard point of wheel carrier to wishbone. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Joints
Wheel carrier - Steering rod (x2) (left/right)	Hard point of wheel carrier to steering rod. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Joints
Vehicle - Stabilizer bar (x2) (left/right)	Hard point of vehicle body to stabilizer. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Joints
Subframe - Stabilizer bar (x2) (left/right)	Hard point of subframe to stabilizer. Can be defined separately (if "Symmetric Suspension" has not been chosen and if "Extended Configuration" is checked)	Joints
Stabilizer bar - Stabilizer link (x2) (left/right)	Hard point of stabilizer bar to stabilizer link. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Joints

## Force Elements

Table 5.18: Definition of the individual suspension components for Four Link MBS.

Name of the element	Comments	Tab
Spring force upper (x2) (left/right)	Hard point of upper spring force. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Force Elements
Spring force lower (x2) (left/right)	Hard point of lower spring force. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Force Elements
Damper force upper (x2) (left/right)	Hard point of upper damper force. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Force Elements
Damper force lower (x2) (left/right)	Hard point of lower damper force. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Force Elements

## Bushings

Table 5.19: Definition of the individual suspension components for Four Link MBS

Name of the bushing	Comments	Tab
Subframe front (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of subframe bushing front. Can be defined separately (if "Symmetric Suspension" has not been chosen and if "Extended Configuration" is checked)	Bushings
Subframe rear left (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of subframe bushing rear. Can be defined separately (if "Symmetric Suspension" has not been chosen and if "Extended Configuration" is checked)	Bushings
Steering gearbox (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of steering gearbox bushing. Can be defined separately (if "Symmetric Suspension" has not been chosen and if "Extended Configuration" is checked)	Bushings
Damper piston (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of damper piston. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Bushings
Wishbone lower front (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of lower wishbone bushing front. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Bushings
Wishbone lower rear (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of lower wishbone bushing rear. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Bushings

Table 5.19: Definition of the individual suspension components for Four Link MBS

Name of the bushing	Comments	Tab
Steering rod (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of steering rod bushing. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Bushings
Steering link (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of steering link bushing. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Bushings

## Mapping

Table 5.20: Definition of the individual suspension components for Four Link MBS

Name of the element	Comments	Tab
Number of grid points	Number of grid points for the tableaus.	Mapping
Maximum compression [m]	Maximum compression which will be used for the tableaus.	Mapping, Kinematics
Maximum steering rack travel [m]	Maximal steering rack travel which will be used for the tableaus.	Mapping, Kinematics
Maximum compression [m]	Maximum compression which will be used for the tableaus.	Mapping, Compliance
Maximum force [N]	Maximum force which will be used for the tableaus.	Mapping, Compliance
Maximum torque [Nm]	Maximum torque which will be used for the tableaus.	Mapping, Compliance

## Misc

Table 5.21: Definition of the individual suspension components for Four Link MBS

Name of the element	Comments	Tab
Static Camber Angle [deg] (x2) (left/right)	Static camber angle. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Misc.
Static Toe Angle [min]	Static toe angle. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Misc.

## 5.11.9 MBS: Twist Beam

This section explains the parameters specific to this MBS model. Please find more information about the general parameters in [section 5.11.5 'Multi Body Suspensions Overview' on page 172](#).

### General

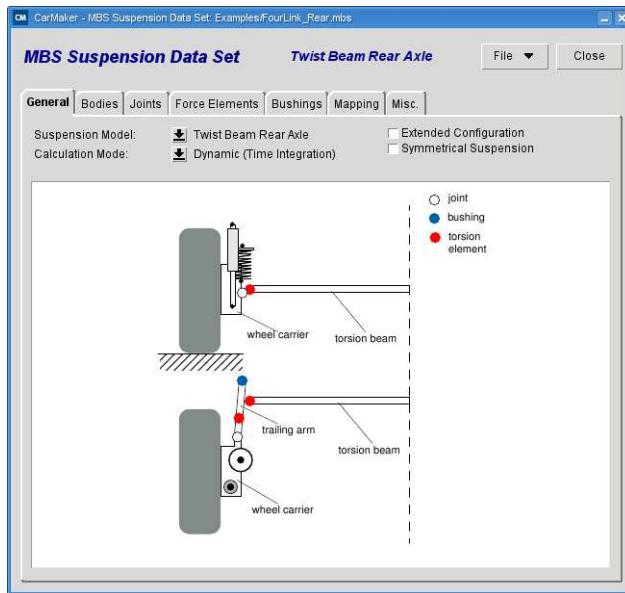


Figure 5.28: Twist beam axle general parameters

- Extended Configuration** If checked the bushing between trailing arm and twist beam is replaced by a torsion element.

### Bodies

Table 5.22: Definition of the individual suspension components for Twist Beam MBS.

Name of the Body	Comments	Tab
Torsion Beam	Measured mass of the torsion beam	Bodies
Wheel Carrier (x2) (left/right)	Measured mass of the Wheel Carrier element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies
Spring (x2) (left/right)	Measured mass of the Damper element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies
Damper (x2) (left/right)	Measured mass of the Damper element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies

Table 5.22: Definition of the individual suspension components for Twist Beam MBS.

Name of the Body	Comments	Tab
Trailing arm (x2) (left/right)	Measured mass of the Trailing arm element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies

## Joints

Table 5.23: Definition of the individual suspension components for Twist Beam MBS.

Name of the joint	Comments	Tab
Origin of the vehicle frame Fr1	The relative coordinates of the frame Fr1 with respect to the Fr <sub>susp</sub> (e.g: x = -3.35m, y = 0.0m, z = -0.3m)	Joints
Wheel carrier - Trailing arm (x2) (left/right)	Hard point of wheel carrier to trailing arm. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Joints
Trailing arm - Torsion beam (x2) (left/right)	Hard point of trailing arm to torsion beam. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Joints

## Force Elements

Table 5.24: Definition of the individual suspension components for Twist Beam MBS.

Name of the element	Comments	Tab
Spring force upper (x2) (left/right)	Hard point of upper spring force. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Force Elements
Spring force lower (x2) (left/right)	Hard point of lower spring force. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Force Elements
Damper force upper (x2) (left/right)	Hard point of upper damper force. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Force Elements
Damper force lower (x2) (left/right)	Hard point of lower damper force. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Force Elements
Torsion beam (x2) (left/right) [Nm/rad]	Spring rates cTx, cTy, cTz for the torsion beam. Can be defined separately (if "Symmetric Suspension" has not been chosen). Only available if "Extended Configuration" is unchecked.	Force Elements

Table 5.24: Definition of the individual suspension components for Twist Beam MBS.

Name of the element	Comments	Tab
Torsion beam (x2) (left/right) [Nms/rad]	Damping rates $k_{Tx}$ , $k_{Ty}$ , $k_{Tz}$ for the torsion beam. Can be defined separately (if "Symmetric Suspension" has not been chosen). Only available if "Extended Configuration" is unchecked.	Force Elements
Trailing arm (x2) (left/ right) [Nm/rad]	Spring rate $c_{Tz}$ for the trailing arm. Can be defined separately (if "Symmetric Suspension" has not been chosen).	Force Elements
Trailing arm (x2) (left/ right) [Nms/rad]	Damping rate $k_{Tz}$ for the trailing arm. Can be defined separately (if "Symmetric Suspension" has not been chosen).	Force Elements

## Bushings

Table 5.25: Definition of the individual suspension components for Twist Beam MBS.

Name of the bushing	Comments	Tab
Trailing arm front (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of trailing arm bushing. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Bushings
Torsion beam (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of torsion beam bushing. Can be defined separately (if "Symmetric Suspension" has not been chosen and only if "Extended Configuration" is checked)	Bushings

## Mapping

Table 5.26: Definition of the individual suspension components for Twist Beam MBS.

Name of the element	Comments	Tab
Number of grid points	Number of grid points for the tableaus.	Mapping
Maximum compression [m]	Maximum compression which will be used for the tableaus.	Mapping, Kinematics
Maximum compression [m]	Maximum compression which will be used for the tableaus.	Mapping, Compliance
Maximum force [N]	Maximum force which will be used for the tableaus.	Mapping, Compliance

Table 5.26: Definition of the individual suspension components for Twist Beam MBS.

Name of the element	Comments	Tab
Maximum torque [Nm]	Maximum torque which will be used for the tabs.	Mapping, Compliance

## Misc

Table 5.27: Definition of the individual suspension components for Twist Beam MBS.

Name of the element	Comments	Tab
Static Camber Angle [deg] (x2) (left/right)	Static camber angle. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Misc.
Static Toe Angle [min]	Static toe angle. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Misc.

## 5.11.10 MBS: Double Wishbone

This section explains the parameters specific to this MBS model. Please find more information about the general parameters in [section 5.11.5 'Multi Body Suspensions Overview' on page 172](#).

### General

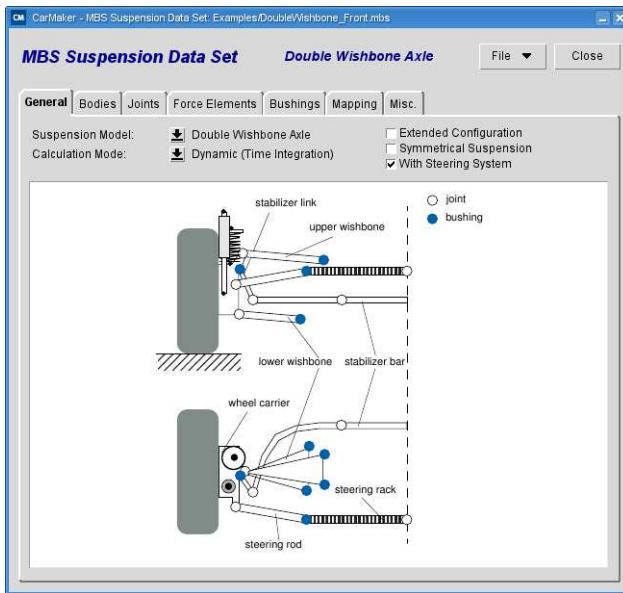


Figure 5.29: Double Wishbone axle general parameters

**Extended Configuration** If checked axle is mounted on a subframe and equipped with a steering system.

**With Steering System** If checked axle is equipped with a steering system.

### Bodies

Table 5.28: Definition of the individual suspension components for Double Wishbone MBS.

Name of the Body	Comments	Tab
Subframe	Measured mass of the Subframe (if "Extended Configuration" has been chosen).	Bodies
Wheel Carrier (x2) (left/right)	Measured mass of the Wheel Carrier element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies
Wishbone lower (x2) (left/right)	Measured mass of the lower Wishbone element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies
Wishbone upper (x2) (left/right)	Measured mass of the upper Wishbone element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies

Table 5.28: Definition of the individual suspension components for Double Wishbone MBS.

Name of the Body	Comments	Tab
Spring (x2) (left/right)	Measured mass of the Damper element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies
Damper (x2) (left/right)	Measured mass of the Damper element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies
Steering gearbox	Measured mass of the steering gearbox element. (if "Extended Configuration" or "With Steering System" has been chosen)	Bodies
Steering rack	Measured mass of the steering rack element. (if "Extended Configuration" or "With Steering System" has been chosen)	Bodies
Steering rod (x2) (left/right)	Measured mass of the steering rod element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies
Stabilizer bar	Measured mass of the stabilizer bar element.	Bodies
Stabilizer link (x2) (left/right)	Measured mass of the stabilizer link element. Masses of the left/right side elements can be provided separately (if "Symmetric Suspension" has not been chosen)	Bodies

## Joints

Table 5.29: Definition of the individual suspension components for Double Wishbone MBS.

Name of the joint	Comments	Tab
Origin of the vehicle frame Fr1	The relative coordinates of the frame Fr1 with respect to the $Fr_{\text{susp}}$ (e.g: $x = -0.75\text{m}$ , $y = 0.0\text{m}$ , $z = -0.3\text{m}$ )	Joints
Wheel carrier - Wishbone lower (x2) (left/right)	Hard point of wheel carrier to lower wishbone. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Joints
Wheel carrier - Wishbone upper (x2) (left/right)	Hard point of wheel carrier to upper wishbone. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Joints
Wheel carrier - Steering rod (x2) (left/right)	Hard point of wheel carrier to steering rod. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Joints
Subframe - Stabilizer bar (x2) (left/right)	Hard point of subframe to stabilizer bar. Can be defined separately (if "Symmetric Suspension" has not been chosen). Available in Extended Configuration	Joints

Table 5.29: Definition of the individual suspension components for Double Wishbone MBS.

Name of the joint	Comments	Tab
Vehicle - Stabilizer bar (x2) (left/right)	Hard point of subframe to stabilizer bar. Can be defined separately (if "Symmetric Suspension" has not been chosen). Available if Extended Configuration is unchecked	Joints
Stabilizer bar - Stabilizer link (x2) (left/right)	Hard point of stabilizer bar to stabilizer link. Can be defined separately (if "Symmetric Suspension" has not been chosen). Available if Extended Configuration is unchecked	Joints

## Force Elements

Table 5.30: Definition of the individual suspension components for Double Wishbone MBS.

Name of the element	Comments	Tab
Spring force upper (x2) (left/right)	Hard point of upper spring force. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Force Elements
Spring force lower (x2) (left/right)	Hard point of lower spring force. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Force Elements
Damper force upper (x2) (left/right)	Hard point of upper damper force. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Force Elements
Damper force lower (x2) (left/right)	Hard point of lower damper force. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Force Elements
Spring force on wheel carrier	If checked then the spring is connected to the wheel carrier if not it is connected to lower wishbone.	Force Elements
Damper force on wheel carrier	If checked then the damper is connected to the wheel carrier if not it is connected to lower wishbone	Force Elements
stabilizer bushing on wheel carrier	If checked then the stabilizer is connected to the wheel carrier. if not it is connected to lower wishbone	Force Elements

## Bushings

Table 5.31: Definition of the individual suspension components for Double Wishbone MBS.

Name of the bushing	Comments	Tab
Subframe front (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of subframe front bushing. Can be defined separately (if "Symmetric Suspension" has not been chosen). Available in Extended Configuration	Bushings
Subframe rear (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of subframe front bushing. Can be defined separately (if "Symmetric Suspension" has not been chosen). Available in Extended Configuration	Bushings
Steering gearbox (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of steering gearbox bushing. Can be defined separately (if "Symmetric Suspension" has not been chosen). Available in Extended Configuration	Bushings
Wishbone lower front (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of Wishbone lower front bushing. Can be defined separately (if "Symmetric Suspension" has not been chosen).	Bushings
Wishbone lower rear (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of Wishbone lower rear bushing. Can be defined separately (if "Symmetric Suspension" has not been chosen).	Bushings
Wishbone upper front (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of Wishbone lower rear bushing. Can be defined separately (if "Symmetric Suspension" has not been chosen).	Bushings
Wishbone upper rear (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of Wishbone lower rear bushing. Can be defined separately (if "Symmetric Suspension" has not been chosen).	Bushings
Steering rod (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of Wishbone lower rear bushing. Can be defined separately (if "Symmetric Suspension" has not been chosen).	Bushings
Stabilizer link (x2) (left/right)	Hard point, orientation and characteristic stiffness in and around xyz of Wishbone lower rear bushing. Can be defined separately (if "Symmetric Suspension" has not been chosen).	Bushings

## Mapping

Table 5.32: Definition of the individual suspension components for Double Wishbone MBS.

Name of the element	Comments	Tab
Number of grid points	Number of grid points for the tableaus.	Mapping
Maximum compression [m]	Maximum compression which will be used for the tableaus.	Mapping, Kinematics
Maximum compression [m]	Maximum compression which will be used for the tableaus.	Mapping, Compliance
Maximum force [N]	Maximum force which will be used for the tableaus.	Mapping, Compliance
Maximum torque [Nm]	Maximum torque which will be used for the tableaus.	Mapping, Compliance

## Misc

Table 5.33: Definition of the individual suspension components for Double Wishbone MBS.

Name of the element	Comments	Tab
Static Camber Angle [deg] (x2) (left/right)	Static camber angle. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Misc.
Static Toe Angle [min]	Static toe angle. Can be defined separately (if "Symmetric Suspension" has not been chosen)	Misc.

## 5.12 Suspension Compliance

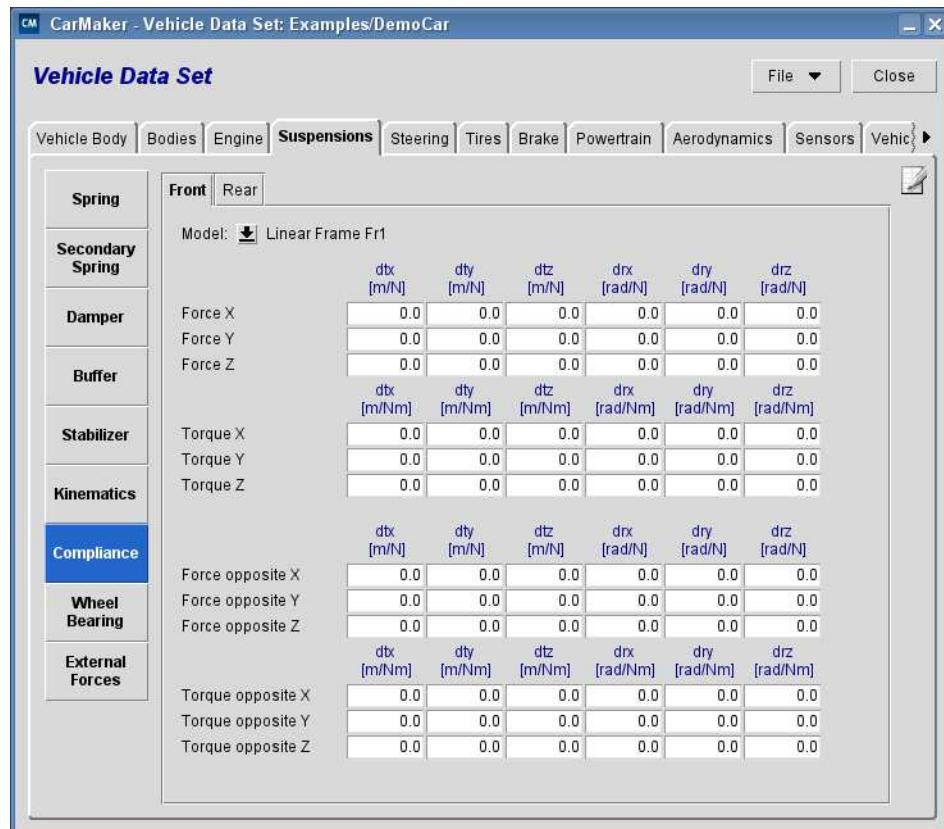


Figure 5.30: Definition of the linear compliance model in CarMaker

The compliance model takes into account the transition of the wheel when forces are applied due to elasticity effects of the suspension.

Defining the compliance in CarMaker is optional, which means you can simulate only with the kinematics and without compliance. Thus, you can define the compliance for 0, 1 or 2 axles.

As for the kinematics, you can choose between various compliance descriptions.

- Model** You can choose between the following compliance descriptions. Some of them are available in the *Model* option of the compliance tab, for others you have to select the mode not specified and define them in an external kinematics file:

Table 5.34: Compliance Models

Mode	Description	Available under Mode
not specified	No compliance is activated, except if it is already defined in the file describing the non-linear kinematics.	yes
Linear Frame Fr1 or Linear Frame Fr2	The compliance displacements caused by forces/torques acting on both wheels are specified by linear coefficients. The forces are given either in <i>Fr1</i> or in <i>Fr2</i> .	yes

Table 5.34: Compliance Models

Mode	Description	Available under Mode
Coeff1DFr1 or Coeff1DFr2	The compliance displacements caused by forces/torques acting on the wheel carrier only are defined by linear coefficients at different wheel compressions. The forces are given either in <i>Fr1</i> or in <i>Fr2</i> .	no, file only
Displace1DFr1 or Displace1DFr2	The compliance displacement caused by forces/torques acting on the current or the opposite wheel carrier are defined by absolute values depending on the forces/torques applied.	no, file only
Displace2DFr1 or Displace2DFr2	The compliance displacement caused by forces/torques acting on the current or the opposite wheel carrier are defined by absolute values depending on the wheel compression and the forces/torques applied.	no, file only



Note that a non-symmetric parameterization is only possible with the compliance models *Coeff1DFr1/2*, *Displace1DFr1/2* and *Displace2DFr1/2*.

#### Definition of Compliance Coefficients for the models Linear Frame Fr1/2 and Coeff1DFr1/2

**Frc.x/y/z**  
**FrcOpp.x/y/z**  
**(N/m) or (N/rad)**

**Trq.x/y/z**  
**TrqOpp.x/y/z**  
**(Nm/m) or (Nm/rad)**

The compliance models *Linear Frame Fr1* or *Linear Frame Fr2* can be parameterized directly in the CarMaker GUI.

For the models *Coeff1DFr1* and *Coeff1DFr2* there is no GUI available. The model parameters are added to the kinematics description file (see section " on page 170 and in the Reference Manual, section "Coef1DFr1 and Coef1DFr2").

However, for both models the definition of the compliance coefficients is the same:

The variables Force\* and Torque\* correspond to the effect of a stress in the associated direction. For instance:

- *Force X* in column *dtx*: effect of a force applied to the wheel center in driving direction (x direction).
- *Torque X* in column *drx*: effect of a torque applied along the x-axis on the camber angle (*rx*).
- *Torque Y* in column *dtx*: effect of a torque applied along the y-axis on the wheel position in the x direction.

The variables Force\_opp\* and Torque\_opp\* represent the effect of a stress on the opposite wheel in the associated direction. For instance:

- *Force opp. X* in column *tx*: effect of a force applied forward on the opposite wheel, on the wheel position in the x direction.
- *Force opp. X* in column *ty*: effect of a force applied forward on the opposite wheel, on the wheel position in the y direction.

Table 5.35: Coefficients for the linear compliance model

Parameter	Description	Unit
dtx	wheel base variation due to force / torque applied	$m_{\text{Wheel travel}}/\text{N}$ Force at wheel center or $m_{\text{Wheel travel}}/\text{Nm}$ Torque at wheel center

Table 5.35: Coefficients for the linear compliance model

Parameter	Description	Unit
dty	track variation due to force / torque applied	$m_{Wheel travel}/N$ Force at wheel center or $m_{Wheel travel}/Nm$ Torque at wheel center
dtz	wheel travel variation due to force / torque applied	$m_{Wheel travel}/N$ Force at wheel center or $m_{Wheel travel}/Nm$ Torque at wheel center
drx	camber variation due to force / torque applied	$rad_{Wheel rotation}/N$ Force at wheel center or $rad_{Wheel rotation}/Nm$ Torque at wheel center
dry	spin angle variation due to force / torque applied	$rad_{Wheel rotation}/N$ Force at wheel center or $rad_{Wheel rotation}/Nm$ Torque at wheel center
drz	toe variation due to force / torque applied	$rad_{Wheel rotation}/N$ Force at wheel center or $rad_{Wheel rotation}/Nm$ Torque at wheel center

### Coeff1DFr1/2, Displace1DFr1/2 and Displace2DFr1/2 compliance models

For both models, the compliance description has to be parameterized in the file chosen for the *MapNL* kinematics mode ([section " on page 170](#)).

In order to know how you can use those modes, please read the Reference Manual, section "Coeff1DFr1 and Coeff1DFr2" and section "Displace1DFr1 and Displace1DFr2" and compare the syntax explained in these chapters to the examples available: in a standard Car-Maker working directory, move to ./Data/Chassis, and open the file *McPherson\_FrontAxle.skc*, the compliance parameterization is at the end.

## 5.13 Suspension: Wheel Bearing Friction

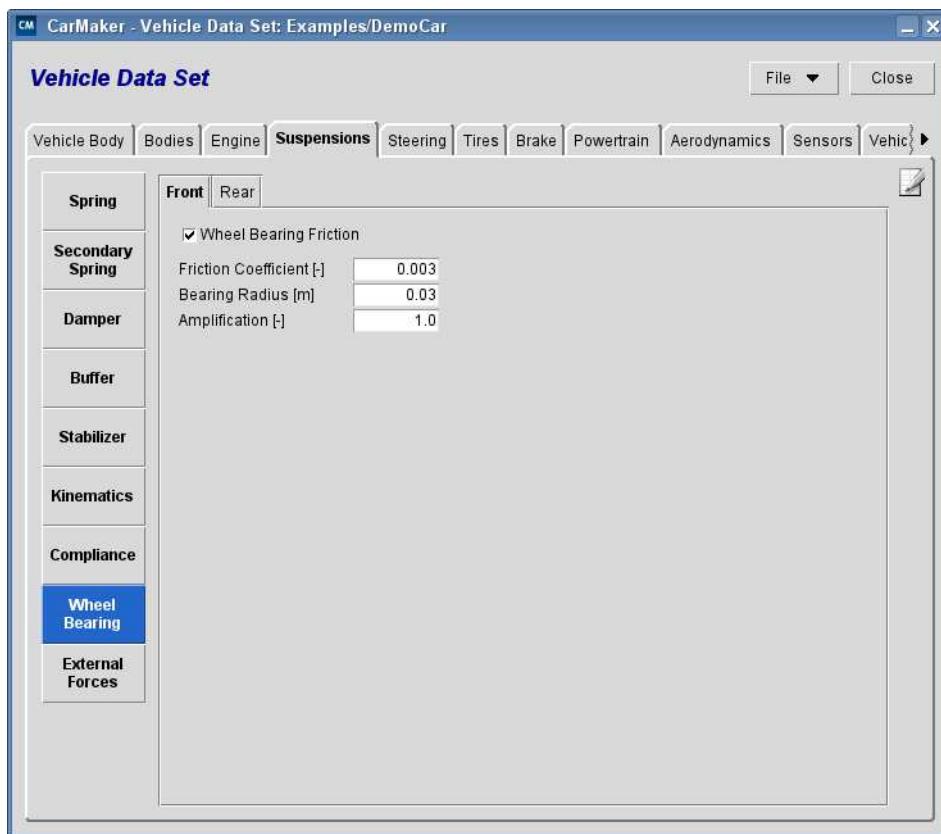


Figure 5.31: Defining a friction torque at the wheel bearings

For a given axle, you have the possibility to take the friction torque at the wheel bearings into account. The friction torque is calculated as follows:

$$T_{Friction} = \mu \cdot F_{Friction} \cdot R = \mu \cdot F_{Axe} \cdot \mu \cdot R$$

- Wheel Bearing Friction** This check box activates/deactivates the effect of friction at the wheel bearings. If activated, other parameters appear.
- Friction Coefficient (-)** The wheel bearing friction is characterized by a single coefficient  $\mu$ , which defines the friction force on the bearing. This value should be delivered by the bearing manufacturer.
- Bearing Radius (m)** The bearing radius  $R$  enables to evaluate the friction torque on the bearing based on the friction force. This parameter can be found out by asking the bearing manufacturer or by measuring it directly on the component.
- Amplification (-)** Using the amplification factor  $amp$  the friction coefficient can be scaled very quickly instead of modifying the parameter itself.

## 5.14 Suspension: External Forces

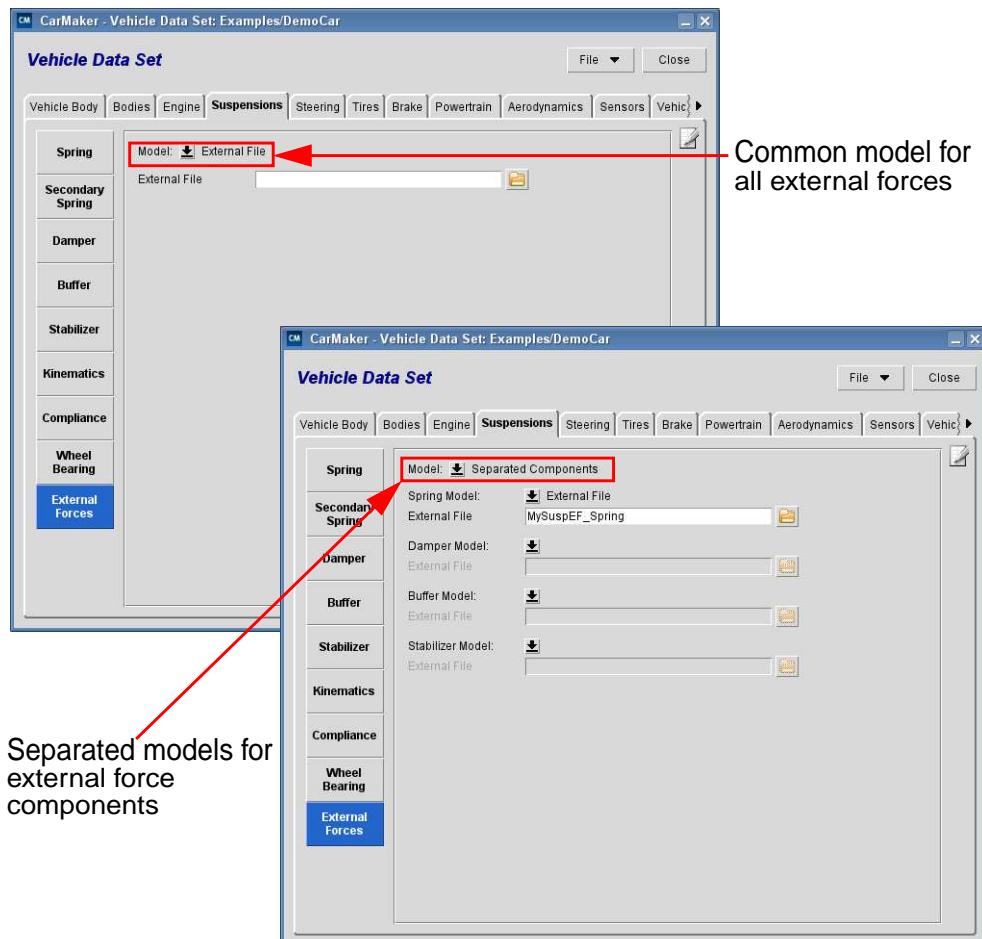


Figure 5.32: Interface for external suspension forces

The external suspension forces interface is designed for specific applications such as adaptive suspension systems. At this interface, forces calculated by an external model can be added to the spring, damper, buffer or stabilizer forces calculated by CarMaker. There are two different model interface kinds available:

- *External File* loads one common model for all external suspension forces of spring, damper, buffer and stabilizer.
- *Separated Components* allows to load up to four different external forces models - one for each kind of force element (spring, damper, buffer and stabilizer).

In both cases, the model can be generated by a c-code or using Matlab Simulink. It may be necessary to read some user specific parameters from files to configure these models. Such parameter files can be selected in the dialog *External File* using the file browser.

Please find further information about how to implement model extensions in the Programmer's Guide.

In the CarMaker for Simulink environment an example model called *Body\_Ctrl.mdl* is provided that shows how external suspension forces can be applied (see the chapter Demonstration examples in the Reference Manual).



## 5.15 Steering System

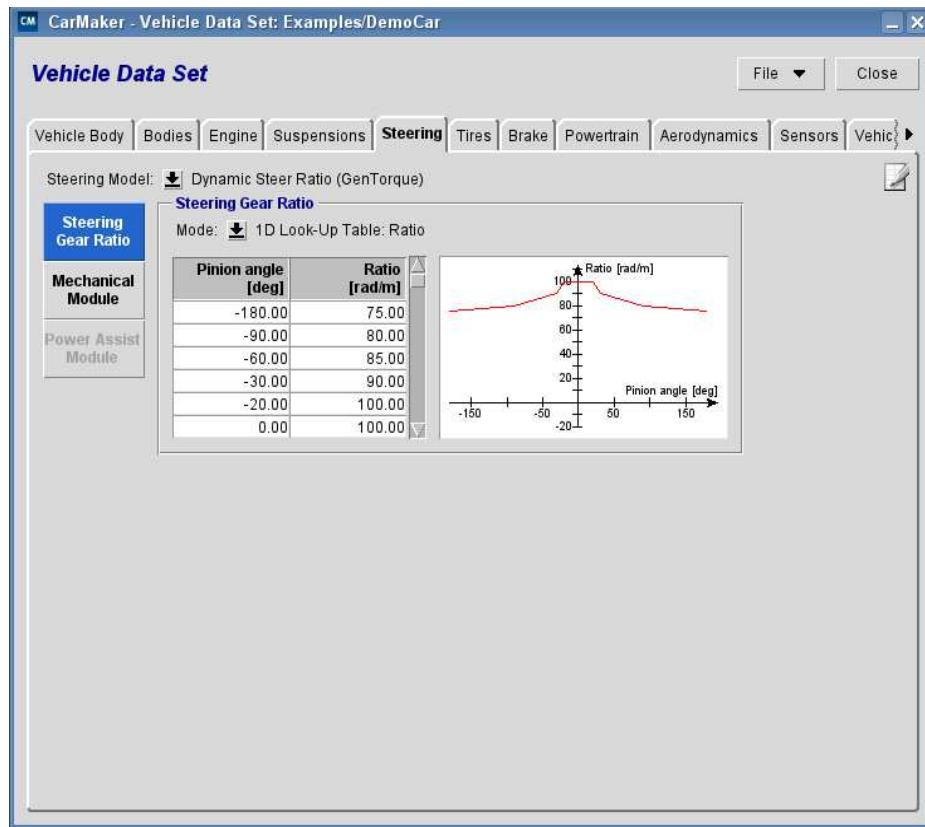


Figure 5.33: The steering tab in the Vehicle Data Set Editor

The steering system consists of a mechanical module which defines the ratio between the steering wheel angle (or steering wheel torque) and the steering rack displacement. Optionally, a power steering unit can be added by selecting the *Pfeffer Steering Model* which also regards friction losses in the steering system. The CarMaker steering model can be replaced by a user's model, too, either using Simulink (see the Simulink examples in the Programmer's Guide) or as c-code model.

### 5.15.1 Steering Models Overview

The following steering models are available: Static Steer Ratio Model

Table 5.36: Steering Models

Model Kinds	Description
Static Steer Ratio	Standard steering model with steering wheel angle as input; no powersteering
Dynamic Steer Ratio	Standard steering model with steering wheel torque as input; no powersteering
External File	Use a user specific steering model
Pfeffer with Power Steering	Extended steering model including powersteering and friction losses
Truck Power Steering	Extended steering model based on 'Pfeffer PowerSteering' for Trucks (Only available in <i>TruckMaker</i> )

The *Static Steer Ratio* steering model expects the steering angle and its derivative as inputs. It includes the calculation of the theoretical steering torque corresponding to the steer angle (observe the quantity *Steer.Whl/TrqStatic* in IPGControl).

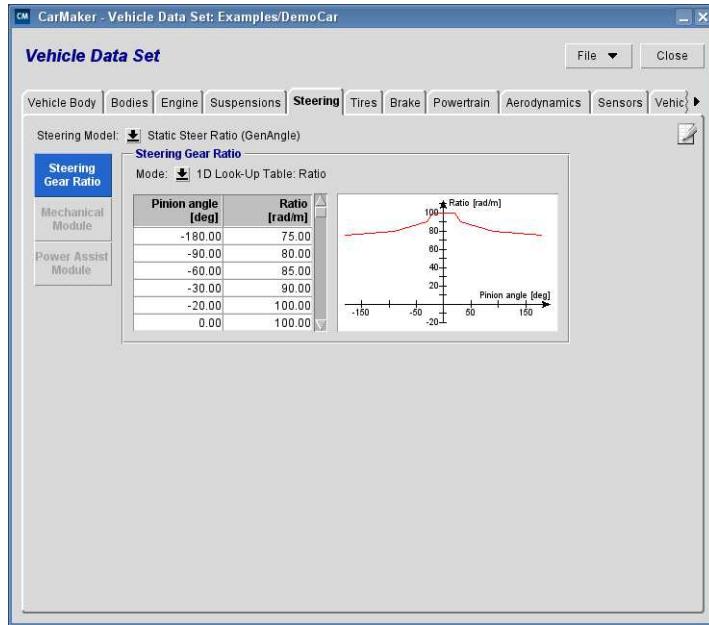


Figure 5.34: Definition of a variable steering gear ratio

### 5.15.2 Static Steer Ratio Model

To define the ratio between the steering wheel angle and the steering rack travel, several options are available:

Table 5.37: Gear ratio modes

Mode	Description	Unit
Characteristic Value	Ratio defined by a constant coefficient of steering wheel angle in [rad] to steering rack travel in [m]	rad/m
Look-Up Table 1D Ratio	Variable ratio with a different coefficient of steering wheel angle in [rad] to steering rack travel in [m] for different steering wheel angles [deg]	rad/m
Look-Up Table 1D Rack Travel	Map, defining the absolute steering rack displacement [m] for various steering wheel angles [deg]	m

Instead of defining the ratio *steering wheel to steering rack travel*, the relation steering wheel to steer angle at the wheel can be parameterized. For this, the following units need to be used:  $\text{rad}_{\text{steering wheel angle}} / \text{rad}_{\text{wheel angle}}$ . In this case it is essential to adapt the kinematics parameterization accordingly: the parameter corresponding to *rz* in the steering column should always be equal to 1, whatever the kinematics mode is.

For more details about the *Static Steer Ratio* model, see the Reference Manual, section "Steer By Angle".

### 5.15.3 Dynamic Steer Ratio Model

The *Dynamic Steer Ratio* model is a steer by torque model. It expects the steering wheel torque as input and the steer angle, velocity and acceleration are returned. The whole steering system is assumed to be stiff

**Steering Gear Ratio** The general transmission ratio between steering wheel angle and steering rack is defined in the same style as for the *Static Steer Ratio* model. Please refer to [section 5.15.2 'Static Steer Ratio Model'](#).

**Mechanical Module** Additionally, the mechanical components used in this steering model need to be parameterized.

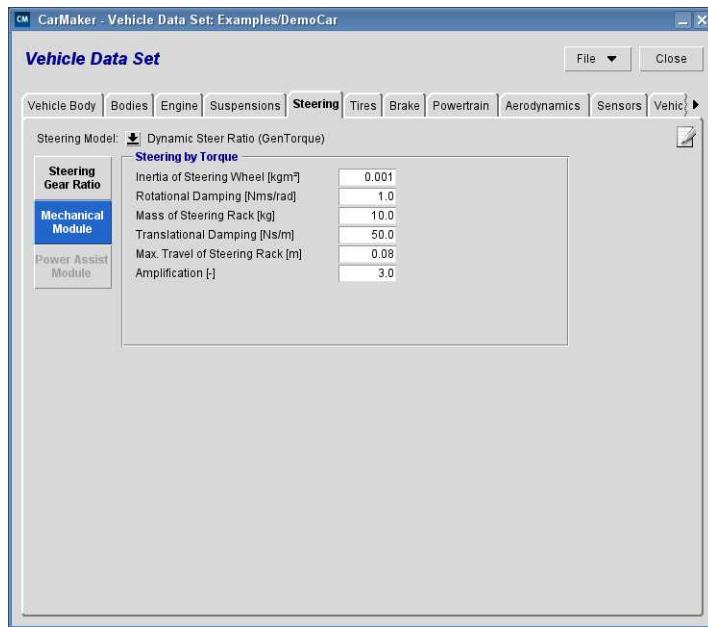


Figure 5.35: Parameterization of the GenTorque steering model

**Steering Gear Ratio** This parameter defines the ratio between the steering rack displacement and the steering wheel angle. The ratio kinds are the same as for the steering model *Static Steer Ratio* (see [Table 5.37: Gear ratio modes](#)).

**Inertia of Steering Wheel (kgm<sup>2</sup>)** The parameter corresponds to the inertia of all rotating parts of the steering system (wheel, column,...). Default value: 0.001kgm<sup>2</sup>.

**Rotational Damping (Nms/rad)** The rotational damping coefficient of the rotating parts (wheel, column, ...) needs to be defined. Default value: 0.1Nms/rad.

**Mass of Steering Rack (kg)** As its name indicates, the mass of the steering rack (in kg) has to be specified in this field. Default value: 10kg.

**Translational Damping (Ns/m)** The translational damping coefficient is the damping coefficient of the steering rack. Default value: 50.0Ns/m.

**Max. Travel of Steering Rack (m)** The parameter specifies the movement range of the steering rack, from center position to stop on one side. Default value: 0.07m.

**Amplification (-)** This parameter specifies the amplification factor of the driver's torque at the steering wheel. The default value is 3.0.

For further information about the *GenTorque* model, see the Reference Manual, section "Steer By Torque".



As an example take the TestRun Examples > CarMakerFunctions > IPGDriver > SteeringByTrq.

## 5.15.4 Pfeffer Power Steering

The Pfeffer Power Steering model is a very detailed model of a steering system including effects caused by friction losses, elasticities and power steering. This model consists of two parts:

- The Mechanical Module, which includes all mechanical components which transfer torque from the steering wheel to the tie rods.
- The Power Assistance Module, which can be of the following modes: hydraulic (HPS), electrical (EPSc and EPSapa) or user defined.

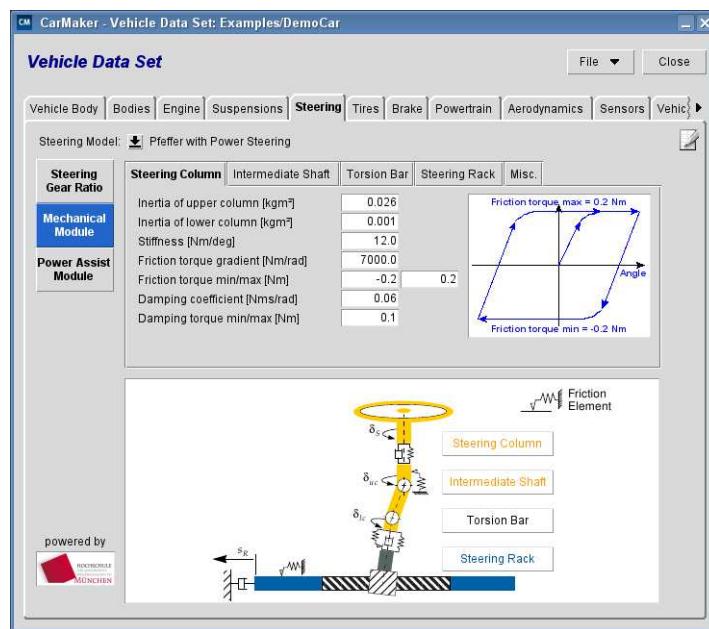


Figure 5.36: Mechanical module of the Pfeffer steering model

### Steering Gear Ratio

Please find more information on the definition of the steering gear ratio in the previous section '[The following steering models are available: Static Steer Ratio Model](#)' on page 202.

### Mechanical Module

This section is divided into the four main components of the mechanical module: steering column, intermediate shaft, torsion bar and steering rack. For each part there are different parameters to specify:

- Steering Column:

Table 5.38: Parameters of the steering column

Parameter	Description	Unit
Total inertia of steering wheel	Inertia of the steering wheel and steering column	kgm <sup>2</sup>
Stiffness	Stiffness coefficient of the steering column	Nm/deg
Friction torque gradient	Exponential spring stiffness of the column friction model	Nm/rad

Table 5.38: Parameters of the steering column

Parameter	Description	Unit
Friction torque min/max	Lower and upper friction torque limit for the steering column	Nm

- Torsion Bar:

Table 5.39: Parameters of the torsion bar

Parameter	Description	Unit
Stiffness	Stiffness coefficient or non-linear stiffness of the torsion bar	Nm/deg
Damping	Damping coefficient of the torsion bar	Nms/rad
Friction torque gradient	Exponential spring stiffness of the torsion bar friction model	Nm/rad
Friction torque min/max	Lower and upper friction torque limit for the torsion bar	Nm
Twist angle min/max	Upper and lower limit of the torsion bar twist angle	deg
Stiffness outside twist limit	Stiffness of the torsion bar outside the twist angle range (outside upper and lower twist angle).	Nm/deg

- Intermediate Shaft:

The lower column angle is related to the upper column angle by a function describing the column non-uniformity due to the double universal joints. In case of selecting none of the following modes, the conventional steering column will be used.

Table 5.40: Intermediate shaft modes

Mode	Description	Unit
1D Look-Up Table	The column non-uniformity is described by a non-linear map containing the lower column angle for each upper column angle.	deg
Double Cardan Joint	The column non-uniformity is described by defining the geometric upper and lower column mount positions in X-Y-Z coordinates. The fork angle describes the difference angle between the two fork planes of the intermediate axis.	m/deg

- Steering Rack:

Table 5.41: Parameters of the steering rack

Parameter	Description	Unit
Mass including steering rods	Mass of the steering rack. Mass of steering rods can be added when it is not included in the body and wheel carrier masses (see Table 5.3: Definition of the extra masses in the Bodies tab)	kg
Max. steering rack travel	Maximum rack travel from zero position to stop on one side. Assumed to be identical on both sides.	m

Table 5.41: Parameters of the steering rack

Parameter	Description	Unit
Friction force gradient	Exponential spring stiffness of the steering rack friction force	N/m
Friction force min/max	Lower and upper friction force limit for the steering rack	N
Friction increase with pressure	Specifies the friction force increase due to pressure. Only used with HPS.	N/bar
Damping coefficient	Damping coefficient of the steering rack	Ns/m
Damping force min/max	Lower and upper damping limit for the steering rack	N

- Misc.:

Table 5.42: Parameters of the Misc. tab

Parameter	Description	Unit
Amplification factor	Amplification factor of all exponential spring friction elements. E.g. used to modify the effects of spring friction as high-frequency excitations from the road are not regarded in CarMaker	-
Stiffness of mesh	Stiffness of the gear / pinion	N/m
Stiffness of hardy disk	Stiffness of the hardy disk	Nm/deg

Please find more information on the meaning of the single parameters and the associated equations in the Reference Manual, section "Steering System Pfeffer".

You can find examples in the folder Examples > CarMakerFunctions > PfefferSteering.



### Power Assist Module

In the *Power Assistance Module* different kinds of power assistance systems are to choose from. Each needs to be configured by different parameters. Please find an overview of all kinds available below. A more detailed description of the models can be taken from the Reference Manual, section "The Power Assistance Module".

As the power steering is a high-frequent system, the calculation of all quantities once in the simulation cycle (1ms) might not be sufficient to gain correct results. For this, the integration time can be enhanced to x-times of one simulation cycle using the option *Integration Sub-steps*.

- Hydraulic Power Steering (HPS):

The hydraulic power steering unit supports the steering wheel motion on the base of hydraulic pressure generated by an engine-driven pump.

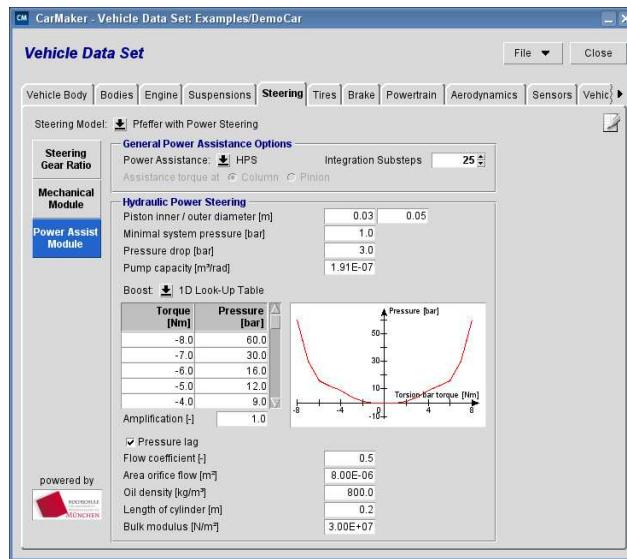


Figure 5.37: Parameters of the HPS module

Table 5.43: Parameters of the Hydraulic power steering module

Parameter	Description	Unit
Piston inner / outer diameter	Inner and outer piston diameter required for the calculation of the assisting force	m
Minimal system pressure	System pressure in the piston without assistance	bar
Pressure drop	HPS pressure loss along the hydraulic line from the cylinder to the pump	bar
Pump capacity	Capacity of the hydraulic pump	m <sup>3</sup> /rad

The *hydraulic boost curve* needs to be specified by a look-up table which can be one- or two-dimensional:

Table 5.44: Specifying the hydraulic boost curve

Boost Modes	Description	Unit
Look-Up Table 1D	Torsion bar torque over target pressure difference between the left and right chamber	Nm, bar
Look-Up Table 2D	Torsion bar torque over target pressure difference between the left and right chamber at given vehicle velocity	Nm, bar, m/s

Optionally, a *pressure lag* can be simulated. When activated, the pressure build-up in the both chambers is delayed by a time lag. The following parameters are available:

Table 5.45: Parameters of the option time lag

Parameter	Description	Unit
Flow coefficient	Flow coefficient to/from chamber	-
Area orifice flow	Area of orifice flow to chamber	m <sup>2</sup>

Table 5.45: Parameters of the option time lag

Parameter	Description	Unit
Oil density	Density of hydraulic oil	kg/m <sup>3</sup>
Length of cylinder	Length of the cylinder	m
Bulk modulus	Bulk modulus of the oil and its containments	N/m <sup>2</sup>

- Electrical Power Steering to Rack (EPS to Rack or EPSSapa):

This system uses an electric motor coupled directly to the steering rack with a belt and ball nut system. It is an axle parallel drive.

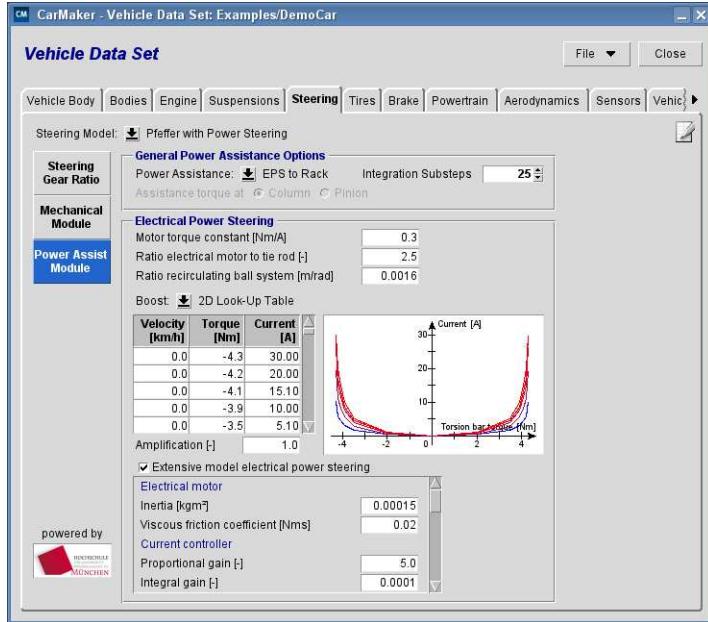


Figure 5.38: Parameters of the ESP to Rack module

Table 5.46: Parameters of the Electrical power steering to rack module

Parameter	Description	Unit
Motor torque constant	Motor torque constant	Nm/A
Ratio electrical motor to tie rod	Transmission ratio electric motor speed / recirculating ball nut speed	-
Ratio recirculating ball system	Transmission ratio of recirculating ball nut system: ball screw translation / ball nut rotation	m/rad

The *electrical boost curve* needs to be specified by a lock-up table which can be one- or two-dimensional:

Table 5.47: Specifying the hydraulic boost curve

Boost Modes	Description	Unit
Look-Up Table 1D	Torsion bar torque over engine current	Nm, A
Look-Up Table 2D	Torsion bar torque over engine current at given vehicle velocity	Nm, A, m/s

- Electrical Power Steering to Column (EPS to Column or EPSc):

This system applies the assisting power generated by an electric motor to the steering column. It can be connected directly to the upper column or to the pinion.

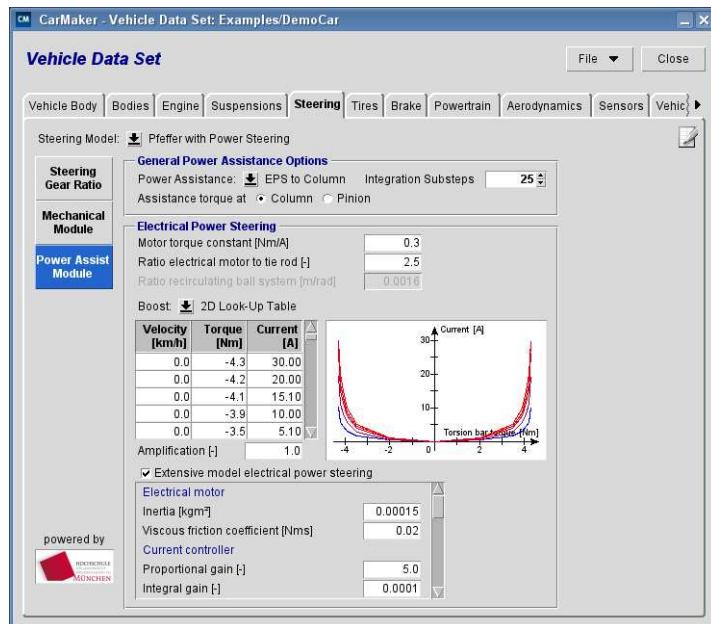


Figure 5.39: Parameters of the EPSc module

Parameter	Description	Unit
Motor torque constant	Motor torque constant	Nm/A
Ratio electrical motor to tie rod	Transmission ratio electric motor speed / recirculating ball nut speed	-
Assistance torque at	Choose between support at steering column or pinion	-

The *electrical boost curve* needs to be specified by a lock-up table which can be one- or two-dimensional:

Table 5.48: Specifying the hydraulic boost curve

Boost Modes	Description	Unit
Look-Up Table 1D	Torsion bar torque over engine current	Nm, A
Look-Up Table 2D	Torsion bar torque over engine current at given vehicle velocity	Nm, A, m/s

- Extensive electrical power steering models:

An extensive model to both electrical power assistance systems is available. As opposed to the simple ESP models, this is a high order model which regards compliance effects in the transmission between engine and column / pinion / rack. It includes an additional degree of freedom for the motor rotation and a controller for the motor current.

Table 5.49: Parameters of the extensive EPS module

Parameter	Description	Unit	Module
Electrical motor:			
Inertia	Inertia of the motor	kgm <sup>2</sup>	EPsap, EPSc

Table 5.49: Parameters of the extensive EPS module

Parameter	Description	Unit	Module
Viscous friction coefficient	Motor viscous friction coefficient	Nms	EPSapa, EPSc
Current controller:			
Proportional gain	Parameter of the current PI controller		EPSapa, EPSc
Integral gain	Parameter of the current PI controller	-	EPSapa, EPSc
Belt:			
Stiffness	Stiffness of the belt	Nm/rad	EPSapa, EPSc
Damping	Damping of the belt	Nms/rad	EPSapa, EPSc
Recirculating ball nut system:			
Mass	Mass of the recirculating ball nut system	kg	EPSapa
Inertia	Inertia of the recirculating ball nut system	kgm <sup>2</sup>	EPSapa
Stiffness	Stiffness of the recirculating balls	Nm/rad	EPSapa
Damping	Damping of the recirculating balls	Ns/m	EPSapa
Rotational friction stiffness	Stiffness of the recirculating ball nut system exponential spring friction model	Nm/rad	EPSapa
Friction torque min/max	Upper and lower limit of the recirculating ball nut system exponential spring friction torque	Nm	EPSapa
Translational friction stiffness	Stiffness of the recirculating ball nut system translational exponential spring friction model	N/m	EPSapa
Friction force min/max	Upper and lower limit of the recirculating ball nut system translational exponential spring friction force	N	EPSapa
Housing:			
Stiffness	Stiffness between the ball nut and housing	N/m	EPSapa
Damping	Damping between the ball nut and housing	Ns/m	EPSapa

- DVA:

Using the DVA mechanism the user can implement his own assistance model (e.g. EHPS) or a steering without assistance. Furthermore, an external assistance force / torque offset can be applied to each power assistance model. The desired power assistance is set using Direct Variable Access on the following quantities:

Table 5.50: Quantities for DVA steering model

Parameter	Description	Unit
Steer.AssistFrc	Assisting force at steering rack	N

Table 5.50: Quantities for DVA steering model

Parameter	Description	Unit
Steer.AssistFrc_Ext	Assisting external force at steering rack	N
Steer.AssistTrqCol	Assisting torque at column	Nm
Steer.AssistTrqCol_Ext	Assisting external torque at column	Nm
Steer.AssistTrqPin	Assisting torque at pinion	Nm
Steer.AssistTrqPin_Ext	Assisting external torque at pinion	Nm

For more information on DVA please see [section 6.3 'DVA: Online Manipulation of the Simulation' on page 357](#).

- User Function:

The desired power assistance is calculated by a user implemented function. Please find an example in the Reference Manual, section "General Parameters for Power Assistance Module".

## 5.15.5 Truck Power Steering

The Truck Power Steering System represents a very detailed model including effects caused by friction losses, elasticities and power steering as well. According to that, the model consists also of the mechanical module and the power assistance module.

The Truck Power Steering System consist of the usual steering column, the intermediate shaft and the torsion bar. Up to this point, the entire system works exactly in the same way as the Pfeffer Power Steering Module. The torsion bar passes the circulating gear box and rotates the pitman arm according to a specified ratio. The pitman arm itself moves the right and left steering knuckles by shifting the steering linkage and tie rod respectively.



Figure 5.40: Animated structure of the Truck Power Steering System

According to structure shown in [Figure 5.40](#), the steering gear ratio is devided into three separate ratios.

**Steering Gear Ratio** The first ratio describes the rotational behavior of the pitman arm related to the steering wheel angle. The second and the third ratio are responsible for the transmission of the pitman arm rotation to the movement of both steering knuckles.

Table 5.51: Steering Gear Ratio modes of the Truck Power Steering System

Parameter	Description	Unit
Characteristic Value	Ratio defined by a constant coefficient of pitman arm angle in [deg] to steering wheel angle in [deg].	deg/deg
Look-Up Table 1D Ratio	Variable ratio with a different coefficient of pitman arm angle in [deg] to steering wheel angle in [deg] for different steering wheel angles [deg].	deg/deg
Look-Up Table 1D Pitman Arm Travel	Map, defining the absolute pitman arm rotation [deg] for various steering wheel angles [deg].	deg

Table 5.52: Steering Linkage Ratio modes of the Truck Power Steering System

Parameter	Description	Unit
Characteristic Value	Ratio defined by a constant coefficient of left and right steering knuckle angle in [deg] to pitman arm angle in [deg].	deg/deg
Look-Up Table 1D Steering Knuckle Ratio	Absolute angles of pitman arm in [deg] to left and right steering knuckle in [deg].	deg

**Mechanical Module** The structure of the mechanical module is quite similar to the Pfeffer Power Steering System. Within the elements steering column, intermediate shaft and torsion bar there are the same parameters available. Only in steering knuckle and misc there are differences.

- Steering Knuckle:

Table 5.53: Parameters of the steering knuckle

Parameter	Description	Unit
Mass	Mass of the steering rack.	kg
Steering knuckle rot. min/max	Minimum and maximum steering knuckle rotation from zero position to the in- resp- outside.	deg
Friction force gradient	Exponential spring stiffness of the steering knuckle friction force	N/rad
Friction force min/max	Lower and upper friction force limit for the steering knuckle	Nm
Damping coefficient tie rod	Damping coefficient of the tie rod	Ns/m
Damping force of the tie rod min/max	Lower and upper damping limit for the tie rod	N
Damping coefficient steering linkage	Damping coefficient of the steering linkage	Ns/m
Damping force of the steering linkage min/max	Lower and upper damping limit for the steering linkage	N

- Misc.:

Table 5.54: Parameters of the Misc. tab

Parameter	Description	Unit
Amplification factor	Amplification factor of all exponential spring friction elements. E.g. used to modify the effects of spring friction as high-frequency excitations from the road are not regarded in CarMaker	-
Stiffness of within steering box	Stiffness within the circulating gear box	N/m
Constant friction at steering knuckle	Specified and constant friction at the steering knuckle. The angles in [deg] between the values should be zero can be specified as well.	Nm
Constant friction at torsion bar	Specified and constant friction at the torsion bar. The angles in [deg] between the values should be zero can be specified as well.	Nm
Constant friction at column	Specified and constant friction at the column. The angles in [deg] between the values should be zero can be specified as well.	Nm

Please find more information on the meaning of the single parameters and the associated equations in the Reference Manual (TruckMaker), section "Truck Power Steering".

### Power Assist Module

The Truck Power Steering System supports only a Hydraulic Power Steering (HPS) Module. This needs to be configured by different parameters. Please find an overview of all kinds available below. A more detailed description of the models can be taken from the Reference Manual (TruckMaker), section "Parameters for Power Assistance Module".

As the power steering is a high-frequent system, the calculation of all quantities once in the simulation cycle (1ms) might not be sufficient to gain correct results. For this, the integration time can be enhanced to x-times of one simulation cycle using the option Integration Substeps.

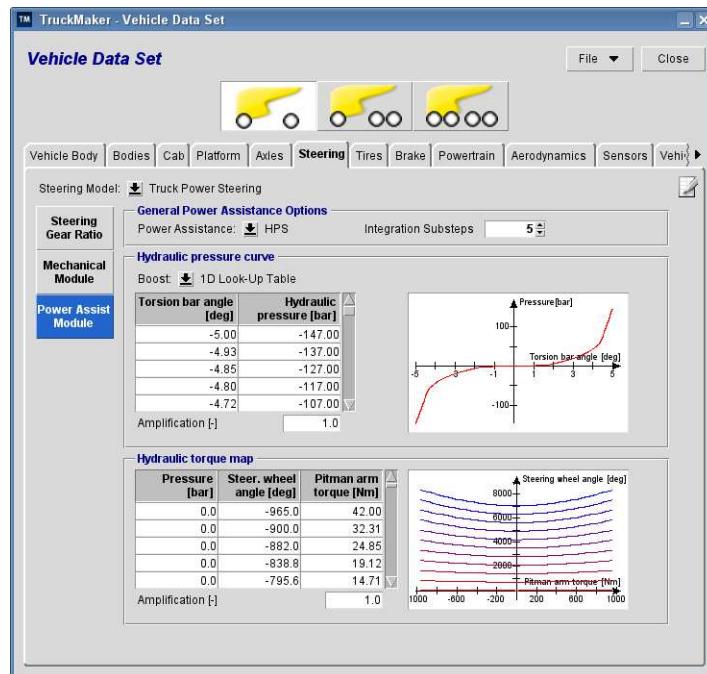


Figure 5.41: Parameters of the HPS module of the Truck Power Steering System

The Hydraulic power steering module consists of the *Hydraulic Pressure Curve* and the *Hydraulic Torque Map*.

The hydraulic boost curve needs to be specified by a look-up table which should be one dimensional. The hydraulic torque map should be two dimensional..

Table 5.55: Specifying the hydraulic curves and maps for the Truck Power Steering System

Parameter	Description	Unit
Look-Up Table 1D Hydraulic Boost Curve	Hydraulic pressure within the circulating ball gearbox in [bar] over the actual torsion bar angle in [deg]	deg, bar
Look-Up Table 2D Hydraulic Torque Map	Pitman arm torque in [Nm] in relation to the hydraulic pressure within the circulating ball gearbox in [bar] and the steering wheel angle in [deg]	bar, deg, Nm

The hydraulic boost curve is also as DVA available.

## 5.16 Brake

The brake system in CarMaker is divided into two parts: The Brake Control unit and the Brake System. Whereas the first is the interface for controllers the brake system represents the physical brake unit.

The pre-implemented brake unit is a hydraulic brake system with a control unit that supports e.g. regenerative braking.

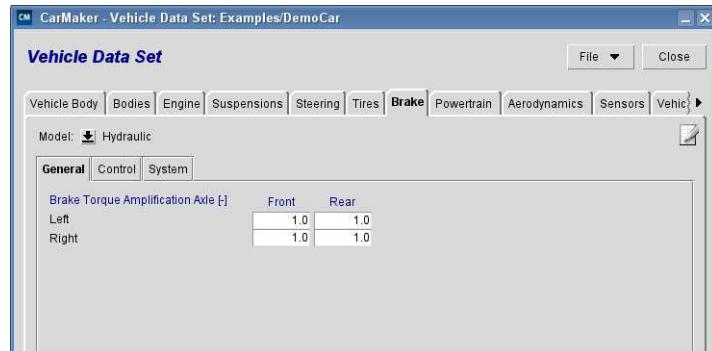


Figure 5.42: General tab in the Brake system configuration

### General

**Brake Torque Amplification (-)** On the *General* tab, the brake ratio for the whole brake system can be easily varied. The brake torque applied to each wheel will be multiplied by this factor.

### 5.16.1 Brake Control (Hydraulic System)

The brake control unit controls the actuation of the hydraulic system. In case of a hybrid or electric vehicle, it also distributes the brake task between the powertrain's electric motors and the hydraulic brake. This way it interacts with the powertrain control unit (see [section 5.20.1 'Parametrizing PT Control'](#)) to ensure reliable regenerative braking.

**Model** There is one controller for hydraulic brake system available:

Table 5.56: Brake Controller Models

Model	Description
Hydraulic Basic Controller	This basic controller for hydraulic brake systems supports several recuperative brake modes. It does not contain any brake assistance system as ABS or ESP.
HIL	Use a real ECU with IO
External File	Use a user specific brake control model that is parameterized via its own infofile

## Model: Hydraulic Basic Controller

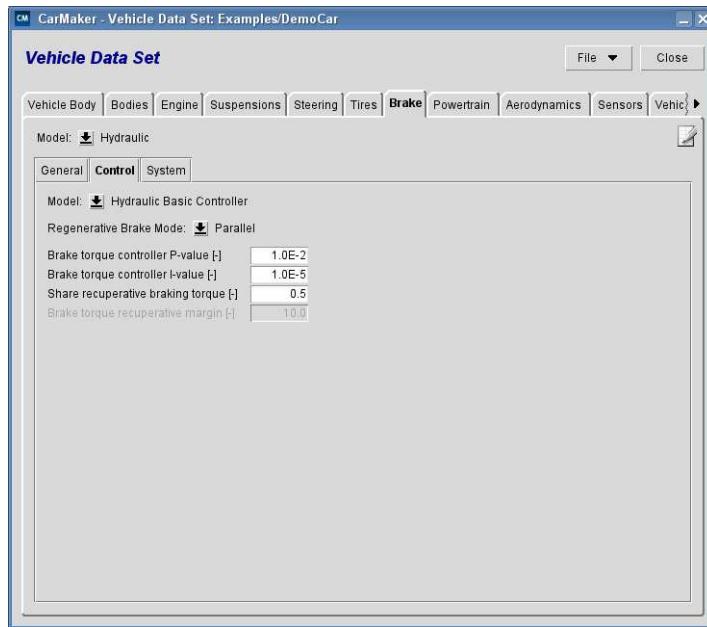


Figure 5.43: Parameters of hydraulic basic brake control

**Regenerative Brake Mode** Here you can select the regenerative brake mode that defines how the target braking torque is distributed between electric motors (used as generator for energy recuperation) and hydraulic brake system:

Table 5.57: Regenerative Brake Modes

Model	Description
No Regeneration	Pure hydraulic braking (e.g. for conventional cars without electric motors)
Parallel	Constant distribution of the brake torque between regenerative braking (electric motors) and hydraulic braking (brake system), e.g. 50% of the demanded brake torque is applied by the electric motors and the other 50% is applied by the hydraulic brake
Serial	The electric motors apply as much of the demanded brake torque as possible (regenerative braking); The hydraulic brake system is used only if the electric motors are not able to satisfy the brake torque demand.

### Regenerative Brake Mode: No Regeneration

This mode has no special parameters.

### Regenerative Brake Mode: Parallel

**Brake torque controller P-/I-value (-)** These parameter define the proportional (P) and integral (I) of the PI controller that controls the hydraulic brake system valves signals.

**Share recuperative braking torque (-)** This factor enables you to set the distribution of regenerative and hydraulic braking torque. E.g. a factor of 0.7 means that 70% of the demanded brake torque is applied by the electric motors and 30% by the hydraulic brake.

### Regenerative Brake Mode: Serial

- Brake torque controller P-/I-value (-)** These parameter define the proportional (P) and integral (I) of the PI controller that controls the hydraulic brake system valves signals.
- Brake torque recuperative margin (-)** If the difference between the target brake torque and the maximum possible regenerative brake torques exceeds this margin value, then the hydraulic brake system is activated to support the regenerative brake torques.

## 5.16.2 Brake System: Hydraulic System

The hydraulic brake system model contains all hydraulic elements of the brake such as master cylinder, valves, pumps.

- Model** At this point it is defined which brake system model is used:

Table 5.58: Brake Models

Model	Description
Pressure Distribution	It is a very simple model with a single virtual master cylinder, without any booster.
External File	Integration of custom made powertrain models
HydESP (External File)	CarMaker already contains a brake model integrated by the External File approach: HydESP is a ESP system hydraulic model, with one master cylinder, booster, pump, low pressure accumulator, valves and pressure loss in the circuits.



Except if you are working on a topic in which the response of the brake system has big effects, you should better use the *Pressure Distribution* model.

### Model: Pressure Distribution

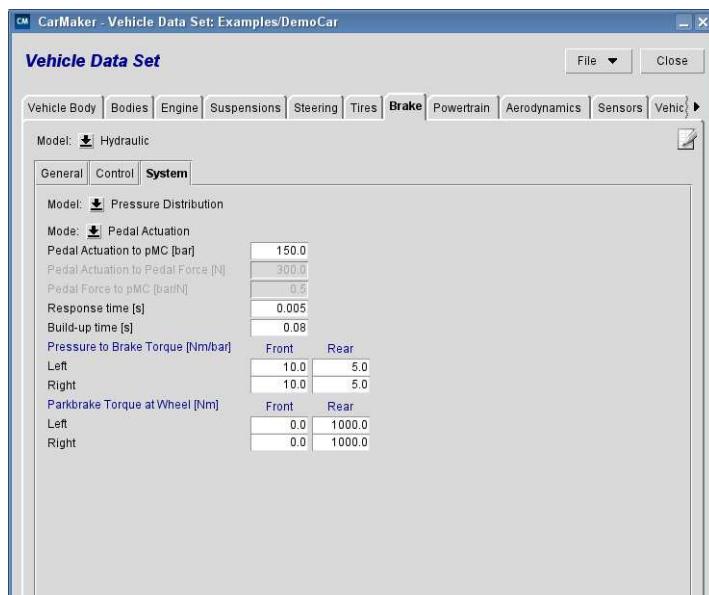


Figure 5.44: Parameters of the hydraulic brake system Pressure Distribution

**Mode** With the *Pressure Distribution* model, there are two ways to control the brake pedal:

Table 5.59: Brake Pedal Transmission Mode

Mode	Description
Pedal Actuation	Here a displacement of the brake pedal (from 0 to 1) is given by the driver.
Pedal Force	Here a force applied on the brake pedal is given.

<b>Pedal Actuation to pMC (bar)</b>	This parameter is used only by the mode <i>Pedal Actuation</i> . It is the ratio of brake pedal position to the corresponding pressure in the master cylinder. E.g. you can enter the maximum pressure in the master cylinder, which means the pressure when the brake pedal is fully pressed.
<b>Pedal Actuation to Pedal Force (N)</b>	This parameter is only used by the mode <i>Pedal Force</i> . It is the ratio of the force the driver applies to the brake pedal and the corresponding pedal movement. E.g. it describes the maximum pedal force that is applicable to the brake pedal as you then divide by one.
<b>Pedal Force to pMC (bar/N)</b>	This parameter is used only by the mode <i>Pedal Force</i> . It is the ratio between the force applied to the brake pedal and the corresponding pressure in the master cylinder.
<b>Response time (s)</b>	Period of time from pressing the pedal till the brake pressure begins to build up.
<b>Build-up time (s)</b>	Period of time from the beginning of the building of brake pressure until it is build up completely.
<b>Pressure to Brake Torque (Nm/bar)</b>	Those parameters describe the ratios between the pressure in the master cylinder and the brake torque applied to each wheel.
<b>Parkbrake Torque at Wheel (Nm)</b>	They describe the brake torque applied to each wheel if the park brake is activated.

### Model: External File

External File is for more specific usage, e.g. if you developed your own brake system model (Power User level).

## Model: External File > HydESP model

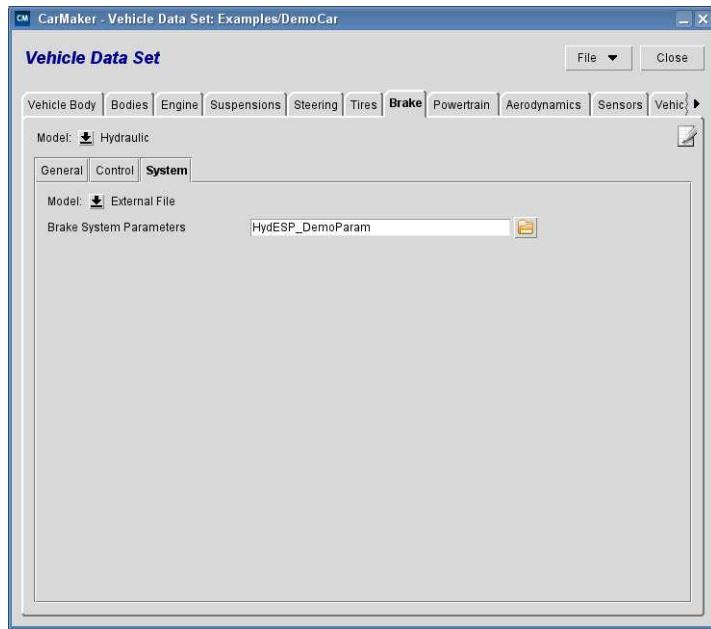


Figure 5.45: Load External File for brake system model HydESP

The HydESP model is a brake system model (without control unit) that has been implemented to CarMaker using the *External File* approach. The model is explained in the example parameterization file *HydESP\_DemoParam* with many comments, but you can also find other explanations in the [Reference Manual](#), section '[Hydraulic Brake System HydESP](#)'.

### Brake System Parameters



To load and parameterize the example ESP hydraulic model, click on select, choose the standard example available (*HydESP\_DemoParam*), and click on edit.

Except for the park brake, you can use alternative units, as long as the final unit of the brake torque at the wheels is in Nm.

## 5.17 Powertrain Overview

In CarMaker, several powertrain configurations are available in order to simulate not only conventional vehicles driven by an internal combustion engine, but also hybrid and electric vehicles. Besides the electro-mechanical components such as engine, electric motors, gearboxes etc. their control units and the vehicle's electric system also are part of the powertrain. Powertrain control (PTControl, also known as hybrid control unit) represents the main control unit that is responsible for the distribution of the demanded torque to the engine and the electric motors.

Please refer to [Reference Manual chapter 'Powertrain'](#) for detailed information about the powertrain module.

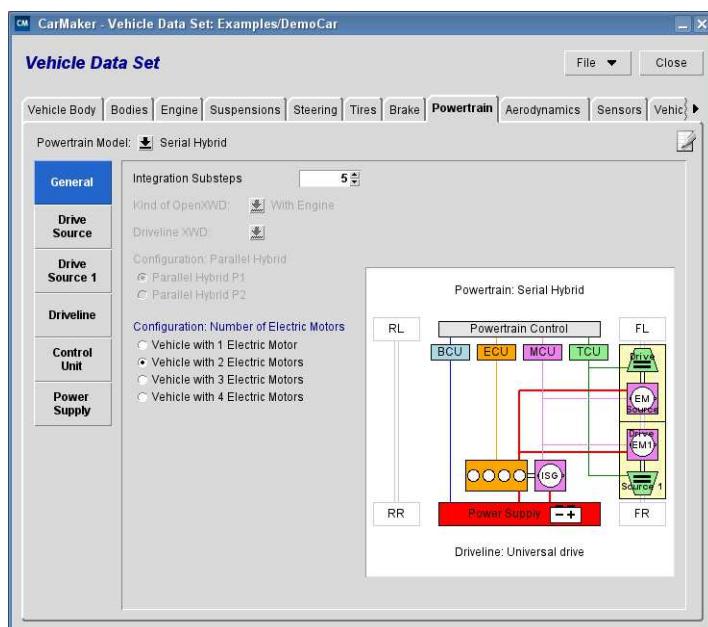


Figure 5.46: The powertrain module

Here the overall architecture of the powertrain is selected. You can choose between ten different configurations as shown in [Table 5.60: Powertrain models](#) and [Figure 5.47](#).

Table 5.60: Powertrain models

Name of the powertrain model	Description
Generic	Conventional powertrain with a combustion engine only
Parallel Hybrid	Hybrid powertrain with a combustion engine and 1-2 electric motors in a parallel configuration
Axle Split Hybrid	Hybrid powertrain also known as <i>parallel-through-the-road</i> ; One axle is driven by a combustion engine, the other axle is driven by 1-2 electric motors
Power Split Hybrid	Hybrid powertrain with a combustion engine and an electric motor that are connected by a planetary gear
Serial Hybrid	Hybrid powertrain also known as <i>range extender</i> ; 1-4 electric motors provide driving torque to the wheels, the combustion engine is used only for electric energy generation
Electrical	Pure electric powertrain with 1-4 electric motors

Table 5.60: Powertrain models

Name of the powertrain model	Description
Open XWD	Free torque distribution to the wheels (e.g. for new hybrid architectures)
AVL Cruise	Interface for co-simulation with AVL Cruise
GT-SUITE	Interface for co-simulation with GT Suite
External File	Integration of custom made powertrain models

### AVL Cruise

This option is a co-simulation of CarMaker and a powertrain model that you have built up in AVL CRUISE. For further specific information about the AVL CRUISE powertrain see [section 10.3 'Powertrain: CarMaker - CRUISE Interface'](#).

### GT Suite

This option is a co-simulation of CarMaker and a powertrain model that you have built up in GT-SUITE. For further specific information about the GT-SUITE powertrain see [section 10.4 'Powertrain: CarMaker - GT-SUITE Interface'](#).

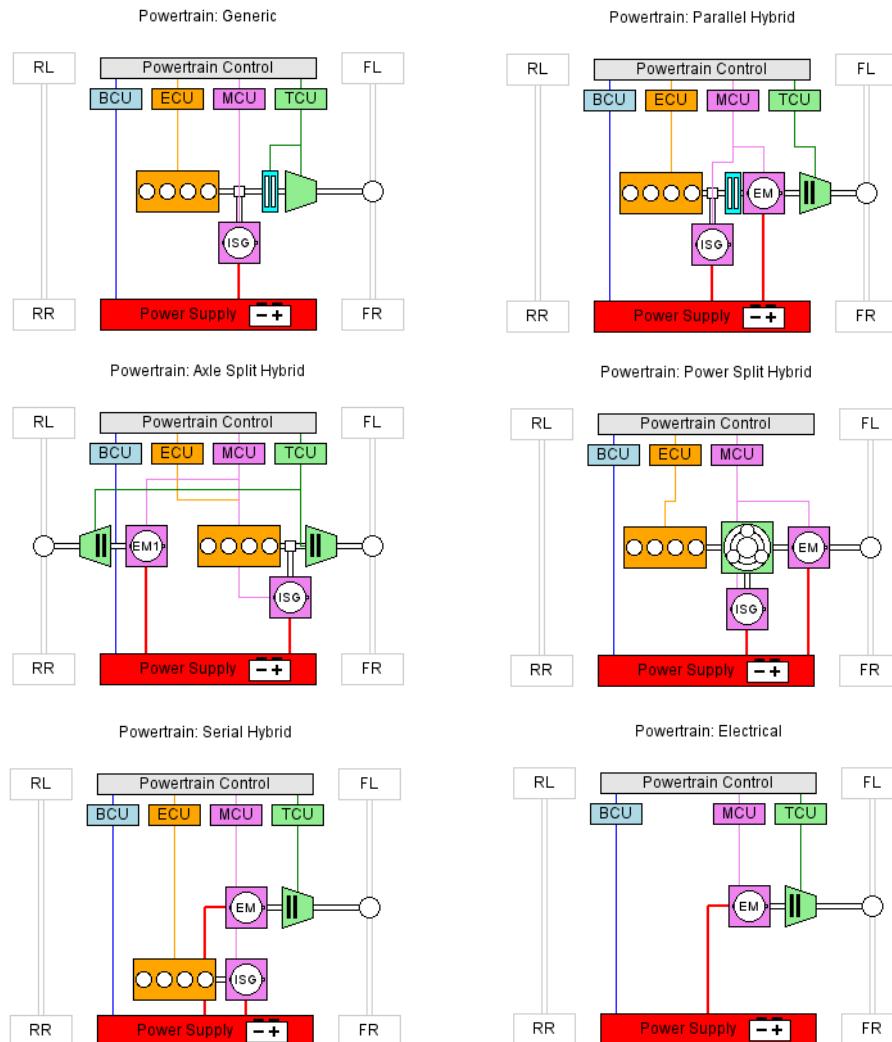


Figure 5.47: Powertrain architectures



Clicking the right mouse button inside the powertrain tab opens a list of predefined powertrain settings that can be useful if you want to start parameterizing a powertrain from scratch. Selecting a predefined setting chooses the configuration of powertrain components needed for this architecture (e.g. corresponding power supply and powertrain control model).

#### Integration Substeps

It enables to calculate the powertrain model more than one time while other models of Car-Maker are only calculated once. This is useful, when the expected frequencies can no longer be sampled with the base sampling rate (1ms) according to the sampling theorem. A value of 5 is recommended for the integrated models when the default sample rate is used.



Please notice that increasing this value directly influences the maximal simulation speed. It is recommended to increase the value in steps of 5 when you observe instabilities in the powertrain system.

#### Configuration Parallel Hybrid

This selection is available if the powertrain model *Parallel Hybrid* is selected only. Here you can switch between the configuration *Parallel Hybrid P1* consisting of combustion engine, integrated starter generator and gearbox and *Parallel Hybrid P2* that contains an additional electric motor and an additional clutch to separate the engine from the rest of the powertrain.

#### Configuration Number of Electric Motors

For the *Axle Split Hybrid* 1 or 2 electric motors can be used for driving on the electric axle. *Serial Hybrid* and *Electrical* powertrains can contain 1-4 electric motors that transfer driving torque to the wheels.

#### Kind of OpenXWD

The powertrain model *OpenXWD* gives the user the possibility to use its own logic for the distribution of drive torques to the wheels. As an advantage compared to the integration of an entirely self-developed powertrain model you do not need to manage the standstill behavior and the integration of the wheel rotation. Control units (e.g. PTControl) for this model kind can be either integrated inside the model or selected from the already existing models.

The distribution of drive torques can be calculated in three manners:

- With Engine: as a function of the gearbox output torque after a combustion engine.
- With Motor: as a function of the gearbox output torque after an electric motor.
- Stand Alone: as a function of a user defined propulsion torque, suitable for alternative driving concepts.

In the first case, the powertrain consists of the same subsystems like the powertrain *Generic* except for the driveline interface. In the second case, the powertrain consists of the same subsystems like the powertrain *Electrical* with one electric motor except for the driveline interface. Using these alternatives, the driveline interface should calculate the propulsion torques to the wheels as a function of the gearbox output.

In the last case, the powertrain only consists of the driveline interface. There is no engine - clutch - gearbox module, so that the propulsion torques to the wheels should be calculated as a function of any user defined torque source.

For detailed information about the OpenXWD powertrain model see [chapter 'Powertrain model Open XWD' in the Reference Manual](#).

## 5.18 Powertrain: Drive Sources

Each powertrain configuration consists of 1-4 *Drive Sources*. Each one is a torque transmission model which transfers a torque from an engine and/or an electric motor to one input shaft of the driveline or one wheel directly. In reaction to that, the wheel speeds are modified and the rotation speed of the different components of the drive source is calculated subsequently.

### 5.18.1 Parametrizing the Engine

The engine is seen as a torque source producing its output depending on the throttle ECU load signal and on its speed (its speed is imposed by the wheel speeds, except for *Serial Hybrid*).

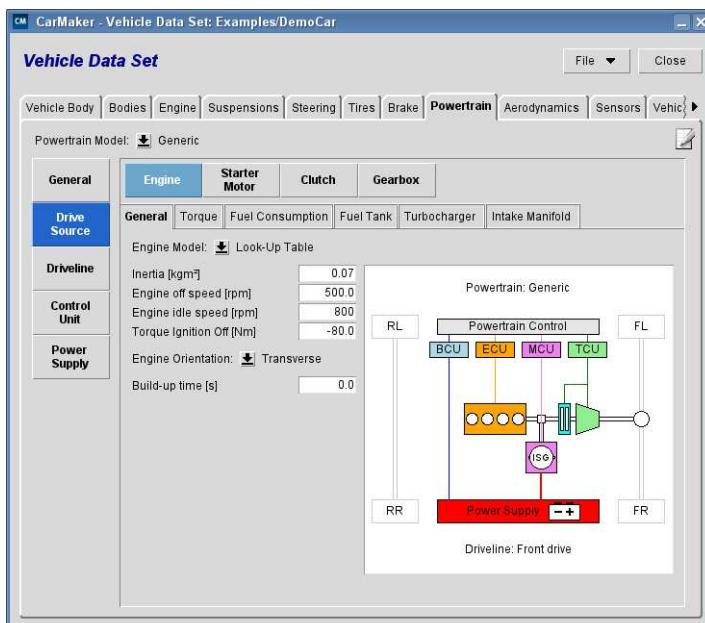


Figure 5.48: Selecting the engine model

**Engine Model** Various engine characteristics are available:

Table 5.61: Engine Models

Model	Description
Characteristic Value	It is a very plain definition of the engine torque: the torque is a linear function of the ECU load signal, but constant over the entire speed range. Many optional parameters are available to tune the engine definition.
Look-Up Table	The definitions of the full load torque as well as of the drag torque are required here. Many optional parameters are available to tune the engine definition.
DVA	No engine definition is parameterized: the engine torque is given directly during the simulation by the user through the DVA interface.
External File	Used to load an engine parameterization file corresponding to another engine model (e.g. Sherpa Engine).

They have some parameters in common.

<b>Inertia (kg m<sup>2</sup>)</b>	The engine inertia sums up all rotating and spinning components of the engine, up to the flywheel. For Ixx it is assumed that the x-axis is along the transmission components.
<b>Idle Speed (rpm)</b>	This parameter specifies the idle speed of the engine. Default value is 800 rpm.
<b>Speed Engine off (rpm)</b>	At this rotation speed the engine is turned off. Default value is 500 rpm.
<b>Torque Ignition off (rpm)</b>	This torque corresponds to the engine drag torque if the ignition is off. If the ignition is on, the drag torque is defined by the engine torque model. Default value is -80 Nm.
<b>Engine Orientation</b>	<p>The engine mounting orientation is needed to take into account the engine propulsion torque that has to be supported by the vehicle body.</p> <ul style="list-style-type: none"> <li>Transverse: The crankshaft is parallel to the vehicle's lateral axle and thus the engine torque is supported by the body around its lateral axis (y).</li> <li>Longitudinal: The crankshaft is parallel to the vehicle's longitudinal axle and thus the engine torque is supported by the body around its longitudinal axis (x).</li> </ul>



The drive torque is usually supported by the body. The part of the drive torque generated by the engine is supported in the joint specified in the tab "Engine" (see [section 5.5 'Engine Mount / Body'](#)). In case the feature *Elastically Mounted Engine* is deactivated, there is no separate support for the engine torque. The drive torque at the wheels is supported in the differential by the body.

## Engine Model: Characteristic Value

The *Characteristic Value* mode should be used only if no exact engine data is available and if the influence of the powertrain is negligible to your simulation results (e.g. steady state circle). Instead of parametrizing the entire engine torque map, only the maximum engine torque at full throttle and the speed range are defined. Out of this data, a linear engine map is interpolated:

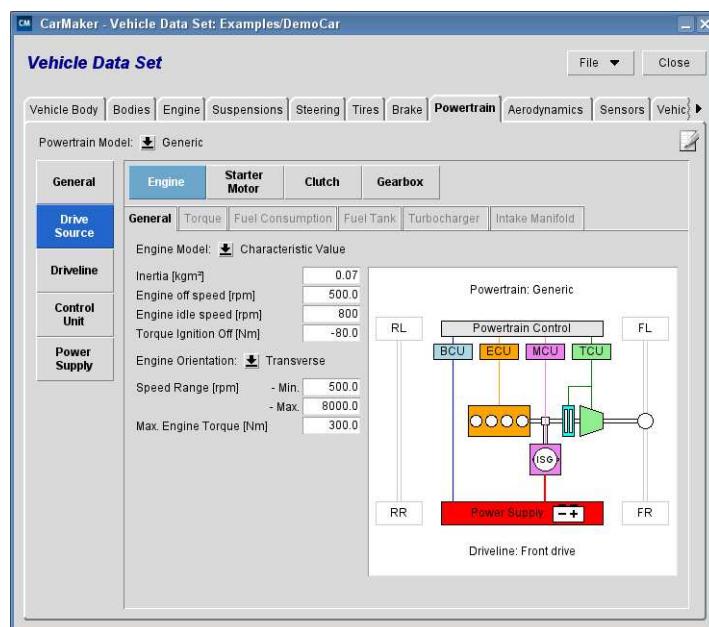


Figure 5.49: Engine mode Characteristic Value

Generally, the engine characteristic depends on the ECU load signal and not on the engine speed as usual. The engine speed is calculated from the engine torque, which is the product of the max. engine torque and the throttle position.

**Speed Range (rpm)** Specifies the minimal and maximal engine rotation speed. E.g. this parameter can be used as an engine speed limiter.

**Max. Engine Torque (Nm)** The maximum engine torque corresponds to the engine output torque at full throttle.

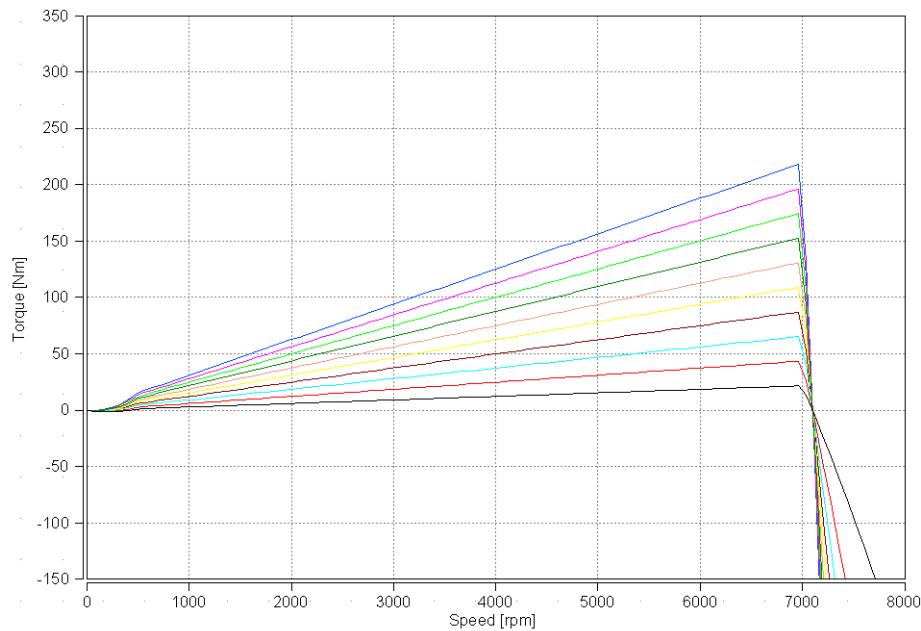


Figure 5.50: Linear engine map

For more information about the engine mode *Characteristic Value* see section 'Engine' in the Reference Manual.

### Engine Model: Look-Up Table

The engine model *Look-Up Table* defines the engine output torque depending not only on the throttle position, but also on its current rotation speed. Thus it is more detailed than the *Characteristic Value* model and needs more parameters.

**Build-up Time (s)** Specifies the build-up time of engine load signal. The delayed build-up is modeled by a PT1 filter and is 0 s by default.

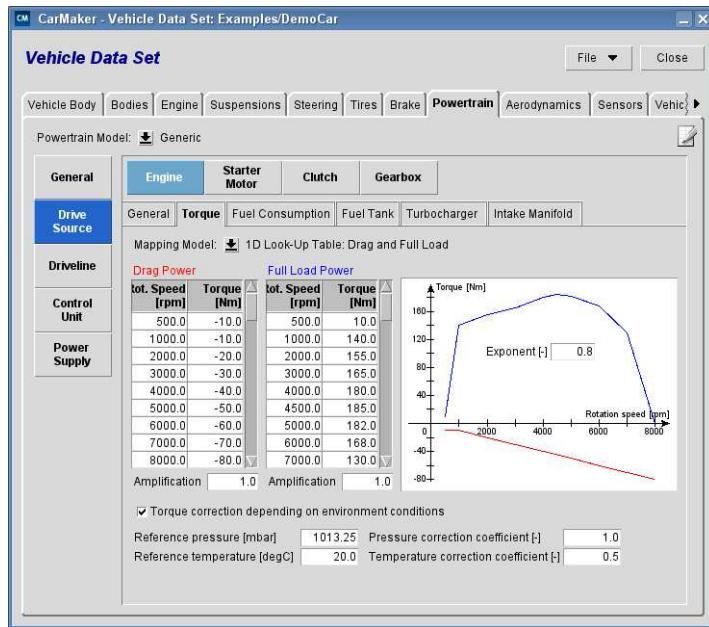


Figure 5.51: Engine mapping for the Look-Up Table 1/2D model

**Mapping Model** In the tab *Torque*, two kinds of look-up tables to describe the engine torque are available:

Table 5.62: Engine Mapping models

Model	Description
Look-Up Table 1D: Drag and Full Load	<p>This is the simplest way to parameterize the torque map and should be used by default.</p> <p>Two characteristics have to be parameterized: the maximum engine torque and the drag torque characteristics. CarMaker interpolates the mid-loads characteristics automatically, depending on the exponent specified.</p>
Look-Up Table 2D: Engine Speed and Gas	<p>This model may be used if you wish to define the torque values at mid-loads on your own. This is only necessary if you have a complete and accurate torque map of an engine and if the tests you want to perform require a very accurate engine response.</p> <p>The parametrization is similar to the model <i>DragFullLoad</i>, but here the engine torque characteristics depends on the Throttle position and engine speed. Thus, some mid-loads characteristics can be defined. Between each mid-load defined by the user, CarMaker automatically interpolates the torque characteristics.</p>



For the mapping model *Engine Speed and Gas*, the engine speed must, as well as the throttle position, always be increasing. The throttle position varies from 0 to 1. For example:

500.0	0.0	-10.0
500.0	0.2	-6.0
500.0	0.4	-2.0
500.0	0.6	2.0
500.0	0.8	6.0
500.0	1.0	10.0
600.0	0.0	0.0
600.0	0.2	8.0
600.0	0.4	16.0
600.0	0.6	24.0

600.0	0.8	32.0
600.0	1.0	40.0
700.0	0.0	5.0
700.0	0.2	18.0
700.0	0.4	31.0
[...]		
7000.0	0.6	50.0
7000.0	0.8	90.0
7000.0	1.0	130.0
8000.0	0.0	-80.0
8000.0	0.2	-64.0
8000.0	0.4	-48.0
8000.0	0.6	-32.0
8000.0	0.8	-16.0
8000.0	1.0	0.0

**Exponent (-)** For the table *Drag and Full Load* an exponent can be set that enables to define the transition of the engine torque from a positive torque value (full load domain) to a negative torque value (drag torque domain). An exponent larger than 1 would correspond to a parabolic transition, an exponent = 1 would be a linear transition, and for an exponent smaller than 1 the change over would be root shaped.

Please find more information on the torque transition function and the usage of the exponent see [section 'Parameter 1D Look-Up Table' in the Reference Manual](#).

**Amplification (-)** This parameter enables to scale the torque characteristics very quickly instead of modifying the entire table. You can also use this parameter to convert the values you have for the torque to the units required by CarMaker. Usually, leave the default value 1.

#### Torque correction depending on environment conditions

It is possible to specify an influence of air temperature and pressure on the engine power and torque. To calculate a correction factor, the following parameters are required:

**Reference pressure (mbar)** Air pressure valid for the entered engine torque table. Default value according to DIN 70020: 1013.25 mbar.

**Reference temperature (degC)** Air temperature valid for the entered engine torque table. Default value according to DIN 70020: 20 degC.

**Pressure correction coefficient (-)** Correction standard for pressure used in engine torque measurement. Default value according to DIN 70020: 1.0.

**Temperature correction coefficient (-)** Correction standard for temperature used in engine torque measurement. Default value according to DIN 70020: 0.5

Please find further information about the formulae of the correction in the Reference Manual.

## Fuel consumption

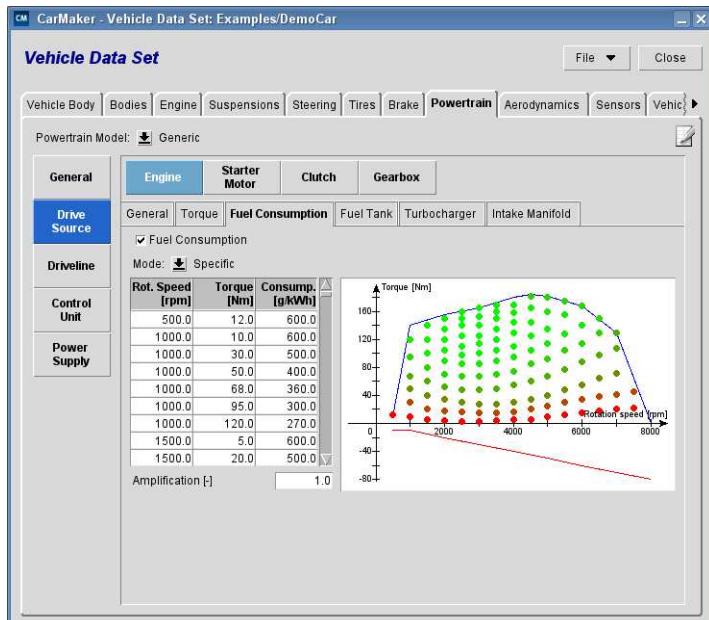


Figure 5.52: Specifying the fuel consumption characteristics

**Fuel Consumption** For the engine model *Look-Up Table*, you can activate the fuel consumption calculation by selecting *Fuel Consumption*.

**Fuel Consumption Mode** The fuel consumption can be stated depending on engine rotation speed and torque. There are two modes:

- Specific: Specify specific fuel consumption in g/kWh
- Absolute: Specify absolute fuel consumption in g/s

**Fuel Consumption Amplification (-)** The amplification factor enables you to vary the fuel consumption map very quickly without changing every single value of the table. You can also use this parameter to convert the values you have for the consumption to the units required by CarMaker. Usually, leave the default value 1.

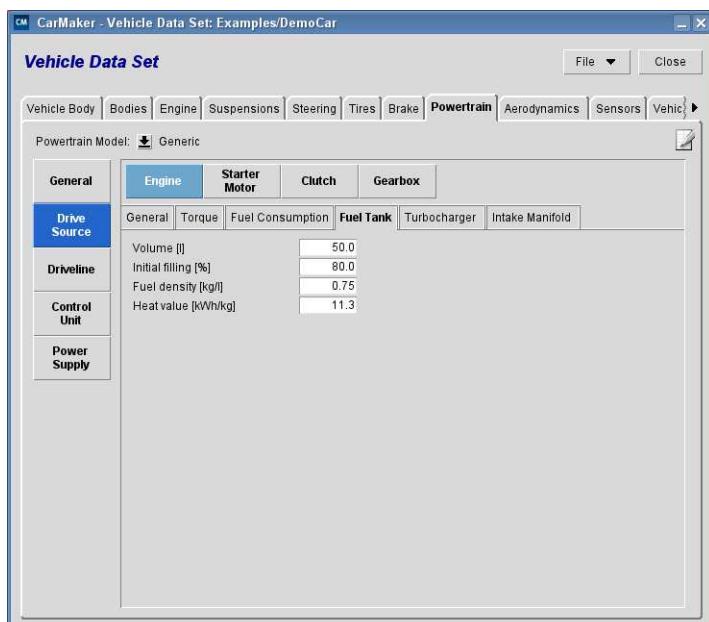


Figure 5.53: Fuel Tank parametrization

The parameters in this tab are related to the vehicle's **fuel tank**. In CarMaker, the fuel is not treated as a (time depending) mass.

- Fuel Tank Volume (l)** This is the global volume of the fuel tank.
- Initial Filling (%)** The initial fuel volume in terms of percentage of the total volume can be defined here.
- Fuel density (kg/l)** This parameter corresponds to the density of the vehicle's fuel.
- Heat value (kWh/kg)** The heat value of the fuel expresses its calorimetic energy.

### Turbocharger

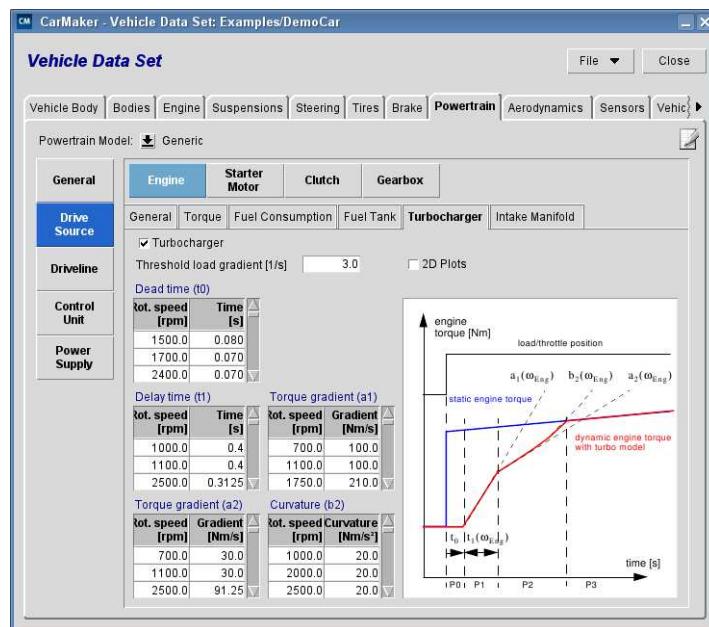


Figure 5.54: Turbocharger parametrization

- Turbocharger** By selecting this tick box, you can activate the turbocharger model. It is only available in combination with the engine model *Look-Up Table*. Without this additional module, the torque calculated by the engine model is delayed only by the build up time. The so called turbo lag cannot be simulated. By activating the turbocharger model (without changing the engine torque map), this effect is added and thus the acceleration behavior changes. For more information about the turbocharger intervention model please refer to [Reference Manual chapter 'Turbocharger'](#).



By selecting and deselecting *2D Plots* you can switch between two different representations of the turbocharger parameters (picture on the right side). So you can choose which one is more useful for application case.

- Threshold load gradient (1/s)** This threshold is compared to the gradient of the ECU load signal. If this signal changes faster than the provided limit, the turbocharger becomes active.
- Dead time (t0)** In this table the delay between load signal and dynamic engine torque reaction in the first phase of turbocharger intervention for different engine rotation speeds.

- Delay time (t1)** The delay time defines the duration of the second phase of turbocharger intervention for different engine rotation speeds.
- Torque gradient (a1)** The torque gradient describes the increasing torque in the second phase of turbocharger intervention (linear function) for different engine rotation speeds.
- Torque gradient (a2)** The torque gradient describes the increasing torque in the third phase of turbocharger intervention (quadratic function, linear part) for different engine rotation speeds.
- Curvature (b2)** The curvature describes the increasing torque in the third phase of turbocharger intervention (quadratic function, quadratic part) for different engine rotation speeds.

### Intake Manifold

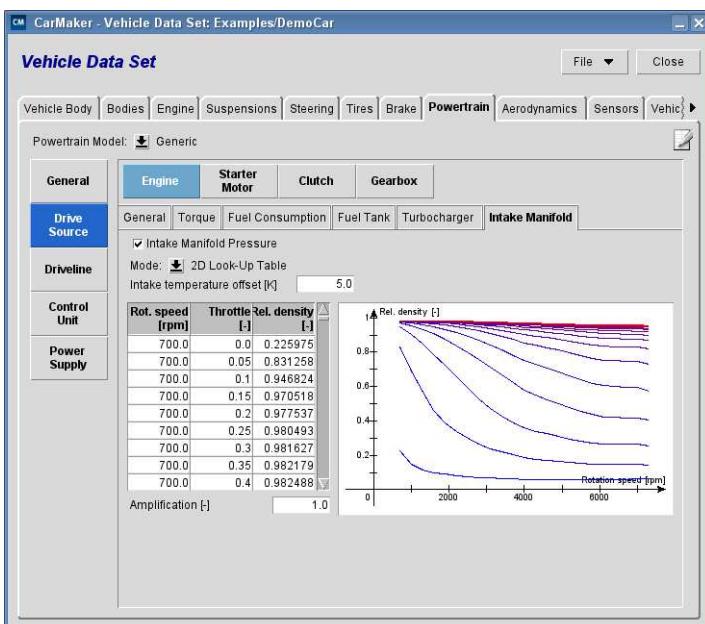


Figure 5.55: Activation of the Intake Manifold pressure model

Here, a simplified model to simulate the intake manifold pressure can be activated. This model can be used e.g. to feed a brake booster. Basically, two modes are available:

- Mode: 2D Look-up Table

With this static model, the intake manifold pressure is calculated depending on the environment air pressure, environment air temperature, the intake manifold temperature and the relative intake manifold density.

#### 2D Look-up Table

The latter can be configured using a 2D look-up table with relative density versus rotation speed and throttle position.

#### Intake temperature offset (K)

Temperature offset in the intake manifold compared to the environment temperature. Default: 5K.

#### Amplification

Amplification factor applied to the output of the intake manifold relative density. Default: 1.

- Mode: Time Integration

This is a dynamical, physical model where the intake manifold pressure is determined by a time integration. It considers the air mass flow, the intake manifold volume a loss coefficient depending on the throttle position, a flow coefficient, the engine speed, engine volume over revolution and the relative filling.

**Intake temperature offset (K)**

Temperature offset in the intake manifold compared to the environment temperature.  
Default: 5K.

**Intake manifold volume (l)**

Volume of the intake manifold. Default: 2.9l.

**Engine cylinder capacity (l)**

Total engine cylinder capacity. Default: 1.8l.

**Ratio cylinder capacity (-)**

Ratio of total engine cylinder capacity used per one revolution. Default: 0.5.

**Area of throttle (cm<sup>2</sup>)**

Area of throttle without losses. Default: 0.5 cm<sup>2</sup>.

Please find further information about the used formulas to both models in the Reference Manual, section "Powertrain > Drive Sources > Engine".

For more information about the whole engine mode "Look-Up Table" please read the [Reference Manual, section' Engine Model Look-Up Table'](#).

## Engine Model: DVA

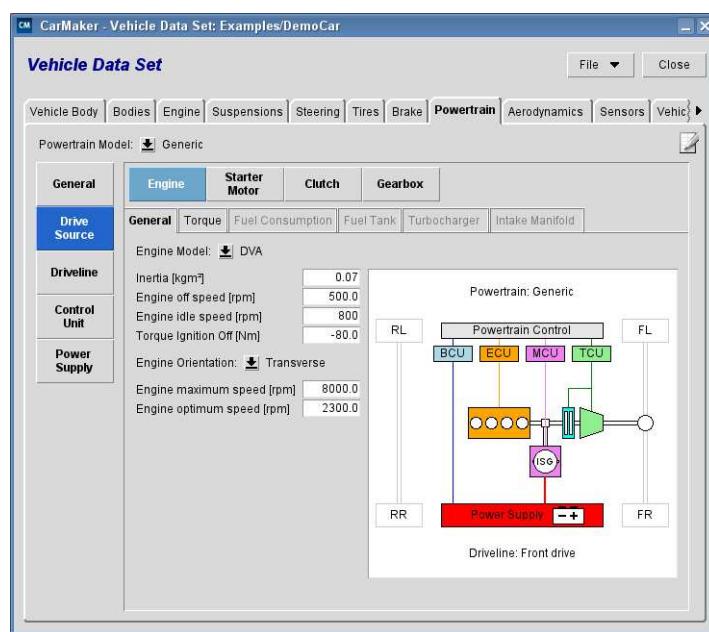


Figure 5.56: Engine model DVA

The DVA engine model is for very specific use. In this mode, the engine torque can be controlled either directly by the user, or by an external model (see Programmer's Guide and [section 6.3 'DVA: Online Manipulation of the Simulation' on page 357](#) in this manual). To modify the engine torque, the User Accessible Quantity `PT.Engine.DVA.Trq` is used, the global engine parameters described above are still available. Additionally you need to provide

some information about the way you calculate the engine torque (e.g. engine maximum speed). These parameters are needed by the control units.

For more information about the engine mode DVA, please read the [Reference Manual section 'Engine Model DVA'](#). More information about the DVA mechanism itself can be found in this document, [section 6.3 on page 357](#).

## Engine Model: External File

External File is for even more specific usage, e.g. if you developed your own engine model (Power User level).

For more information about the engine model, please read the [Reference Manual section 'Engine'](#).

### 5.18.2 Parametrizing the Starter Motor

The starter motor (integrated starter generator, ISG) provides a torque to the powertrain e.g. in order to start up the engine depending on the load signal provided by the MCU. It is directly applied to the engines output shaft (except powertrain architecture *Power split Hybrid*).

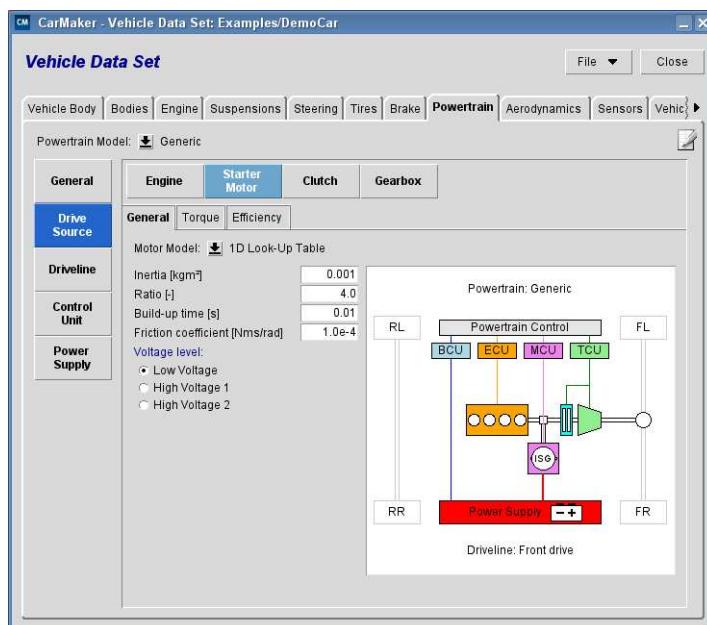


Figure 5.57: Selecting the starter motor model

**Motor Model** Different starter motor models are available:

Table 5.63: Starter Motor Models

Model	Description
Starter	This is a simple starter model that only acts as a motor (can not be used as generator)
1D Look-Up Table	This model includes motor and generator characteristics of the starter motor. There are different levels of complexity available.

They have some parameters in common.

- Inertia ( $\text{kgm}^2$ )** The starter motor inertia sums up all rotating and spinning components of the motor. For  $I_{xx}$  it is assumed that the x-axis is along the transmission components.
- Ratio (-)** This parameter describes a constant transmission ratio between the starter motor output shaft and the powertrain shaft that it is connected to (usually engine output shaft).
- Voltage level** The voltage level specifies to which electric circuit of the power supply the starter motor is connected. The electric power consumed or generated by the starter motor is automatically taken into account in the power balance of the selected circuit. There are up to three circuits available depending on the power supply model selected (please refer to [section 5.21 'Powertrain: Power Supply'](#))
  - Low Voltage: electric circuit of the low voltage battery
  - High Voltage 1: electric circuit of the high voltage battery
  - High Voltage 2: separated high voltage circuit without direct connection to a battery

## Motor Model: Starter

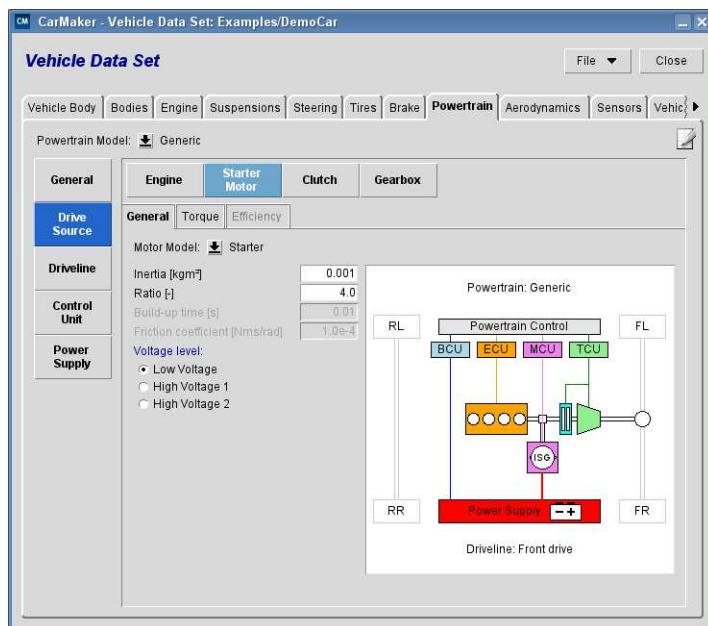


Figure 5.58: Starter Motor Model Starter parametrization

The motor model *Starter* can be used as motor only. It has no generator mode. Thus its main application area are conventional powertrains. The maximum torque provided is defined by three parameters.

- Mechanical power (kW)** This parameter defines the maximum mechanical power that the starter motor is able to provide.
- Maximum torque (Nm)** This corresponds to the maximum torque provided by the starter motor.
- Maximum rot. speed (rpm)** This parameter contains the maximum motor rotation speed.

## Motor Model: 1D Look-Up Table

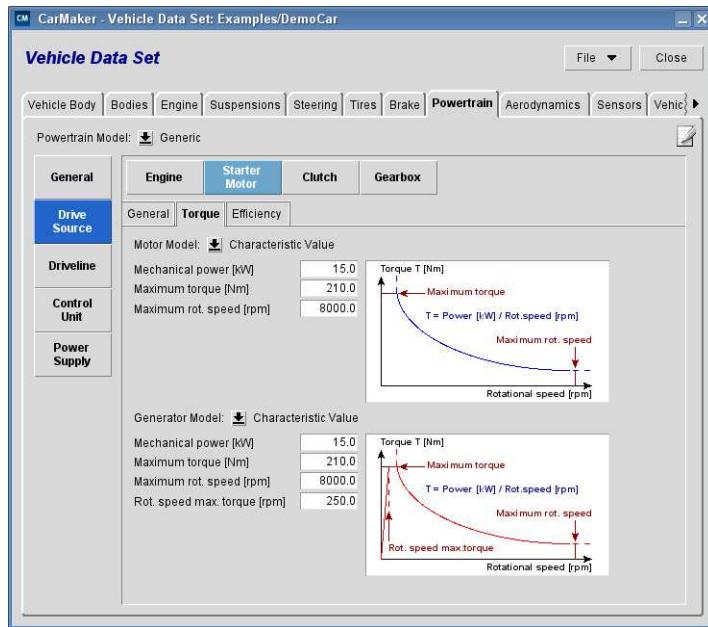


Figure 5.59: Starter Motor Model 1D Look-Up Table parametrization

The starter motor model *Mapping* is an electric motor model with motor and generator characteristics. It is equal to the electric motor model *Mapping*. Please refer to [section 'Motor Model: 1D Look-Up Table' on page 236](#) for a list of parameters.

### 5.18.3 Parametrizing the Electric Motor

The electric motor is seen as a torque source producing its output depending on the throttle MCU load signal and on its speed. In motor mode (conversion of electric power into mechanic power) the sign of rotation speed and torque are equal. In generator mode (conversion of Mechanic power into electric power) rotation speed and torque have different signs.

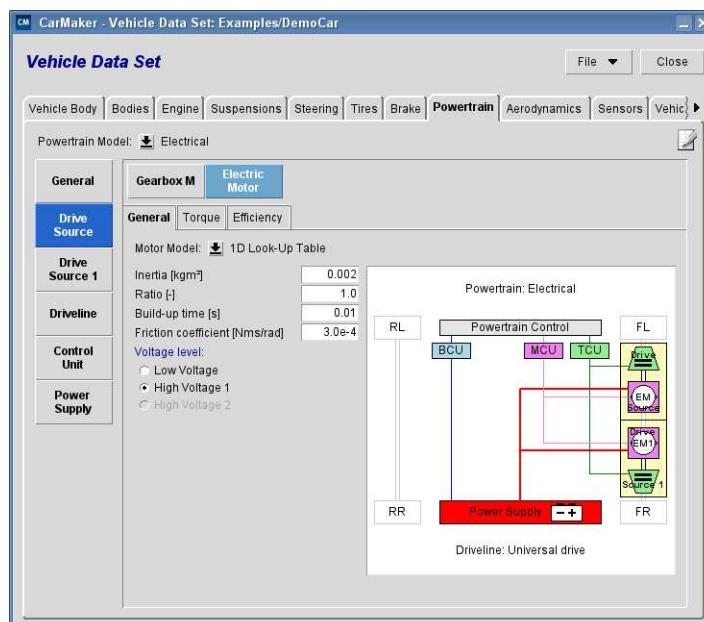


Figure 5.60: Selecting a Motor model

Hybrid and electric powertrains maybe contain several electric motors. In this case, they are numbered starting with motor, motor 1, motor 2, etc. These names are used to differentiate their User Accessible Quantities and CarMaker for Simulink interface signals, too.

**Motor Model** Currently, one electric motor model is available:

Table 5.64: Starter Motor Models

Model	Description
1D Look-Up Table	This model includes motor and generator characteristics of the electric motor. There are different levels of complexity available.

## Motor Model: 1D Look-Up Table

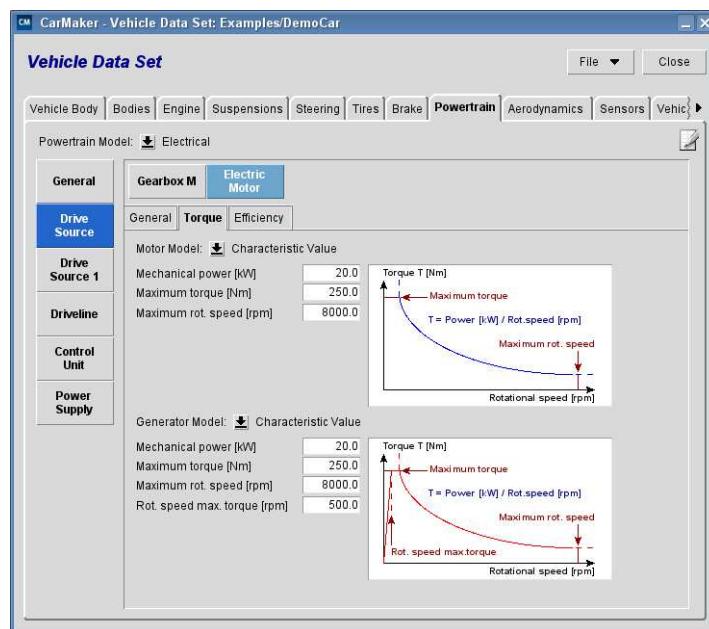


Figure 5.61: Motor Model 1D Look-Up Table parametrization

- Inertia ( $\text{kgm}^2$ )** The electric motor inertia sums up all rotating and spinning components of the motor. For  $I_{xx}$  it is assumed that the x-axis is along the transmission components.
- Ratio (-)** This parameter describes a constant transmission ratio between the electric motor output shaft and the driven shaft that it is connected to.
- Build-up time (s)** Specifies the build-up time of the load signal. The delayed build-up is modeled as a PT1 filter.
- Friction coefficient (Nm/rad)** The friction coefficient defines the internal motor friction torque that acts as a drag torque if the motor load is zero.
- Voltage level** The voltage level specifies to which electric circuit of the power supply the electric motor is connected. The electric power consumed or generated by the starter motor is automatically taken into account in the power balance of the selected circuit. There are up to three circuits available depending on the power supply model selected (please refer to section 5.21 'Powertrain: Power Supply').
- Low Voltage: electric circuit of the low voltage battery

- High Voltage 1: electric circuit of the high voltage battery
- High Voltage 2: separated high voltage circuit without direct connection to a battery

### Torque Motor Model/ Generator Model

Using the electric motor model Mapping, the electric motor's torque characteristics (as a function of its rotation speed) are defined by two curves

- the maximum motor torque (available if the load signal is 1)
- the maximum generator torque (available if the load signal is -1)

Both can be defined independently by two ways:

Table 5.65: Motor Torque Models

Model	Description
Characteristic Value	The torque characteristic is described by three (motor) or four (generator) characteristic values
1D Look-Up Table	The torque characteristic is provided by a table containing the maximum available torque for different rotation speeds.

### Torque Motor Model: Characteristic Value

For parameterizing this model, you only need to know basic specific values of the electric motor that are usually provided in motor data sheets. They are sufficient to define the torque curve as shown in [Figure 5.62](#).

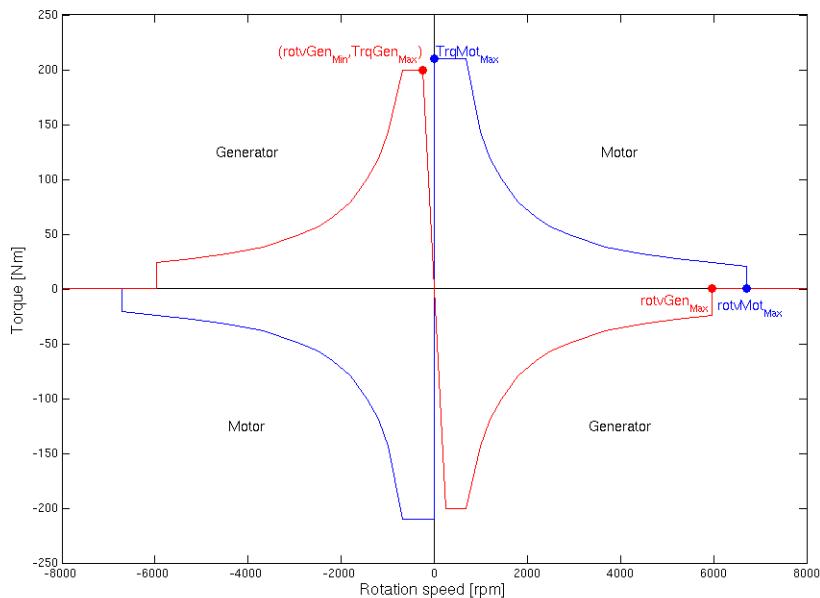


Figure 5.62: Motor torque model characteristic value

**Mechanical power (kW)** The electric motor's mechanical power in motor/generator mode influences the curvature of the motor/generator torque curve.

**Maximum torque (Nm)** The overall maximum motor/generator torque corresponds to the torque available for small motor rotation speeds.

**Maximum rot. speed (rpm)** The parameter defines the maximum allowed motor rotation speed for motor/generator mode.

- Rot. speed max. torque (rpm)** This parameter is available for generator mode only. Unlike the motor torque, the maximum generator torque is not available for very small rotation speeds, but only if the rotation speed passes the limit provided by this parameter.

### Torque Motor Model: 1D Look-Up Table

In the motor torque model *1D Look-Up Table* the maximum motor/generator torque is described by its values for different rotation speeds. CarMaker interpolates the mid-loads characteristics automatically. You can use this model e.g. if the form of your motor's torque characteristic is different to the one displayed in [Figure 5.62](#).

- Amplification (-)** This parameter enables to scale the torque characteristics very quickly instead of modifying the entire table. You can also use this parameter to convert the values you have for the torque to the units required by CarMaker. Usually, leave the default value 1.

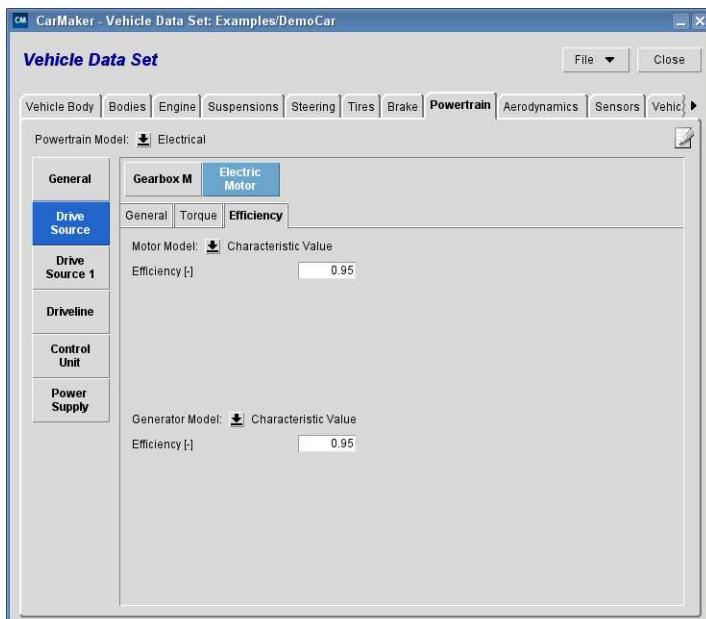


Figure 5.63: Selecting the motor efficiency model

- Efficiency Motor Model/Generator Model** The efficiency of the electric motor can be defined for motor and generator mode separately. For both, two efficiency models are available.

Table 5.66: Motor Efficiency Models

Model	Description
Characteristic Value	The efficiency is considered as a constant factor.
2D Look-Up Table	The efficiency ( $\eta$ ) depends on the normalized rotation speed (current rot. speed/maximum rot. speed) and normalized torque (current torque/maximum torque) of the motor/generator mode.

- Efficiency (-)** This parameter corresponds to the electric motors overall efficiency in motor/generator mode. It is available for the efficiency motor model *Characteristic Value*.
- Amplification (-)** This factor applies to the column  $\eta$  of the *2D Look-Up Table* model. It enables you to change motor efficiencies in a very quick way.

## 5.18.4 Parametrizing the Gearbox

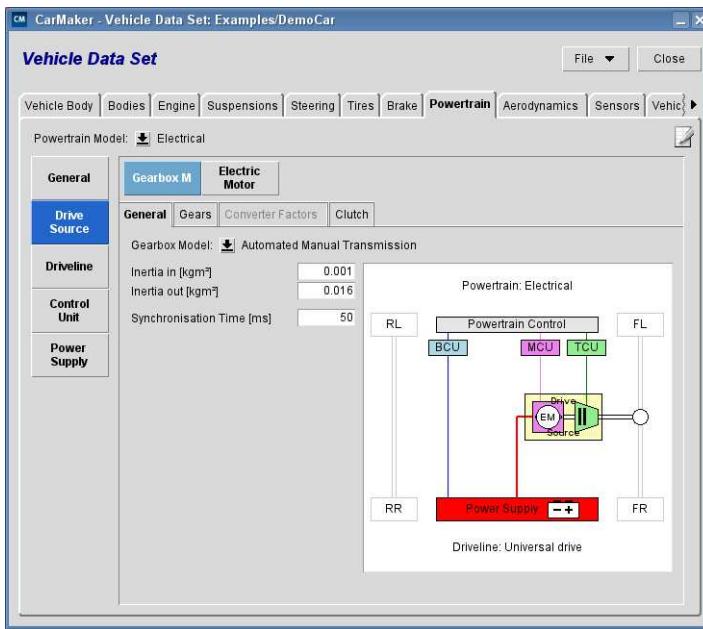


Figure 5.64: Selecting a Gearbox model

The gearbox module enables to convert the rotation speed of the shafts as well as the transferred torque just like a real gear box would do. Since CarMaker 5.0 the gearbox model includes its clutch.

There can be several gearboxes in hybrid and electric powertrains. In this case, the following naming convention is used:

- If the gearbox belongs to the same Drive Source as the engine, it is called *Gearbox*.
- If the gearbox belongs to a Drive Source without engine, it is called *Gearbox M*.
- If there are several Drive Sources with electric motors only, the motor's number is added to the gearbox name (e.g. *Gearbox M1* if it is in the same Drive Source as *Motor 1*).

**Gearbox Model** You have the choice between six gearbox modes:

Table 5.67: Gearbox Modes

Mode	Description
Manual	This mode corresponds to a manual gearbox without integrated clutch model. It can only be used for <i>Generic</i> powertrains in combination with the stand alone clutch model <i>Friction!</i>
Automatic	This mode corresponds to an automatic gearbox without integrated clutch model. It can only be used for <i>Generic</i> powertrains in combination with the stand alone clutch model <i>Converter!</i>
Automatic with Converter	This mode contains an automatic gearbox with a hydraulic converter (see section 'Clutch Model: Converter') as internal clutch model. Gears are shifted via the TCU.
Automated Manual Transmission	This mode contains an automatic gearbox with a friction clutch (see section 'Clutch Model: Converter') as internal clutch model. Gears are shifted via the TCU.

Table 5.67: Gearbox Modes

Mode	Description
DVA	No gearbox ratios are parameterized: the transmission ratio is given directly during the simulation by the user through the DVA interface.
No Gearbox	The engine or motor output shaft is directly to the driveline without transmission.
External File	Used to load a gearbox parameterization file corresponding to an user gearbox model.

All gearbox modes (except *No Gearbox*) have some parameters in common.

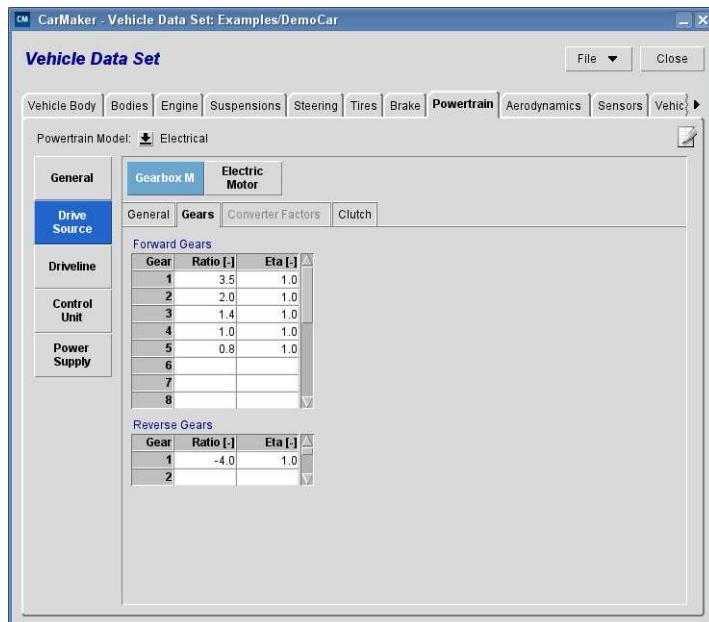


Figure 5.65: Gearbox parametrization

- Inertia in ( $\text{kgm}^2$ )** This sums up the inertia of all rotating and spinning gearbox input components, e.g. input shaft, assuming that the x-axis is along the transmission components for  $I_{xx}$ .
- Inertia out ( $\text{kgm}^2$ )** This sums up the inertia of all rotating and spinning gearbox output components, e.g. the intermediate and output shafts, assuming that the x-axis is along the transmission components for  $I_{xx}$ .
- Synchronization time (ms)** This parameter sets the time required to synchronize the input and output shaft speeds during gear change. Default: 50 ms
- Forward Gears/  
Reverse Gears** In the tab *Gears* you can set the transmission ratios (*Ratio*) and efficiencies (*Eta*) for each forward and backward gears. Each ratio  $I_n$  is calculated as follows:  $I_n = T_{\text{out}} / T_{\text{in}}$ .  $T_{\text{out}}$  being the output torque (after the gear box), and  $T_{\text{in}}$  the input torque (ahead of the gear box, after the clutch). The last efficiency value in the list is valid for all remaining gears.
-  If the ratio is not provided for a gear, it means that this gear does not exist. Thus you can define a gearbox without reverse gears by not providing any ratio in the lower table.

## Gearbox Model: Automatic with Converter

### Inertia Converter in ( $\text{kgm}^2$ )

This inertia sums up all rotating and spinning clutch input components, from the flywheel up to the first clutch plate, assuming that the x-axis is along the transmission components for  $I_{xx}$ .

### Inertia Converter out ( $\text{kgm}^2$ )

This inertia sums up all rotating and spinning clutch output components, from the second clutch plate up to the entry to the gear box, assuming that the x-axis is along the transmission components for  $I_{xx}$ .

### Converter Factors

The parameters set under *Converter Factors* describe the hydraulic converter which is used as an internal clutch for this gearbox mode. Please refer to section 'Clutch Model: Converter' for a detailed description of these parameters.

## Gearbox Model: Automated Manual Transmission

### Clutch

The parameters set in the tab *Clutch* describe the friction clutch model which is used as an internal clutch for this gearbox mode. Please refer to section 'Clutch Model: Friction' for a detailed description of these parameters.

## Gearbox Model: DVA



Figure 5.66: Gearbox Model DVA parametrization

The DVA gearbox model is for very specific use. In this mode, the gearbox transmission ratio can be controlled either directly by the user, or by an external model (see Programmer's Guide and section 6.3 'DVA: Online Manipulation of the Simulation' on page 357 in this manual). To modify the transmission ratio, the User Accessible Quantity `PT.GearBox.DVA.i` (in case of a Drive Source without engine, replace `GearBox` by the gearbox name). This model still needs some of the global gearbox parameters described above. Additionally you need to provide some information about the way you calculate the transmission ratio. These parameters are needed by the control units.

### Number of Forward/Reverse Gears

Information about the number of different forward/reverse ratios available.

**Ratio of Forward/  
Reverse Gear 1**

Information about the transmission ratio of the first gear.

For more information about the gearbox mode DVA, please read the [Reference Manual section 'Gearbox Model DVA'](#). More information about the DVA mechanism itself can be found in this document, [section 6.3 on page 357](#).

### Gearbox Model: No Gearbox

This model connects the engine or motor output shaft to the driveline directly and so it has no parameters. It can be used e.g. for electric vehicles with wheel hub motors. There is no TCU model required for this gearbox kind.

### Gearbox Model: External File

*External File* is for even more specific usage, e.g. if you developed your own gearbox model (Power User level).

## 5.18.5 Parametrizing the Clutch

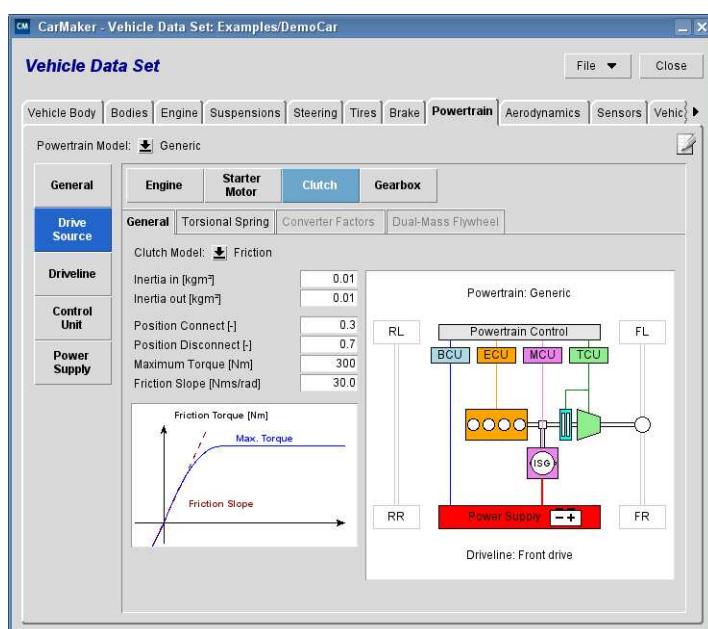


Figure 5.67: Selecting a clutch model

The clutch model enables to transfer all or none of the torque produced by the engine or electric motor to the drive line part. In CarMaker, clutches usually are submodels of a gearbox. Only the powertrain architectures *Generic* and *Parallel* may contain a stand alone clutch. The User Accessible Quantities for a stand alone clutch simply start with *Clutch*, for internal clutches that contain <name of gearbox>.Clutch.

**Clutch Model** The following clutch models are available:

Table 5.68: Clutch Models

Mode	Description
Friction	Model of a conventional mechanical friction clutch (used with gearbox mode <i>Manual</i> or inside <i>Automated Manual Transmission</i> ).
Converter	Model of a hydraulic converter that is used as a clutch (with gearbox mode <i>Automatic</i> or inside <i>Automatic with Converter</i> )

Table 5.68: Clutch Models

Mode	Description
Closed	This is not a real clutch model, but replaces the clutch by a simple shaft (e.g. for powertrain architecture <i>Parallel P1</i> )
Dual-Mass Flywheel	Model of a dual-mass flywheel clutch (used with gearbox model <i>Manual</i> )
DVA	In this model, no transmission torque is calculated for the clutch. This torque is given directly during the simulation by the user through the DVA interface.
External File	Used to load a clutch parameterization file corresponding to an user clutch model.

These models have two parameters in common (except the model *Closed* that needs no parameters).

**Inertia in ( $\text{kgm}^2$ )** This inertia sums up all rotating and spinning clutch input components, from the flywheel up to the first clutch plate, assuming that the x-axis is along the transmission components for  $I_{xx}$ .

**Inertia out ( $\text{kgm}^2$ )** This inertia sums up all rotating and spinning clutch output components, from the second clutch plate up to the entry to the gear box, assuming that the x-axis is along the transmission components for  $I_{xx}$

## Clutch Model: Friction

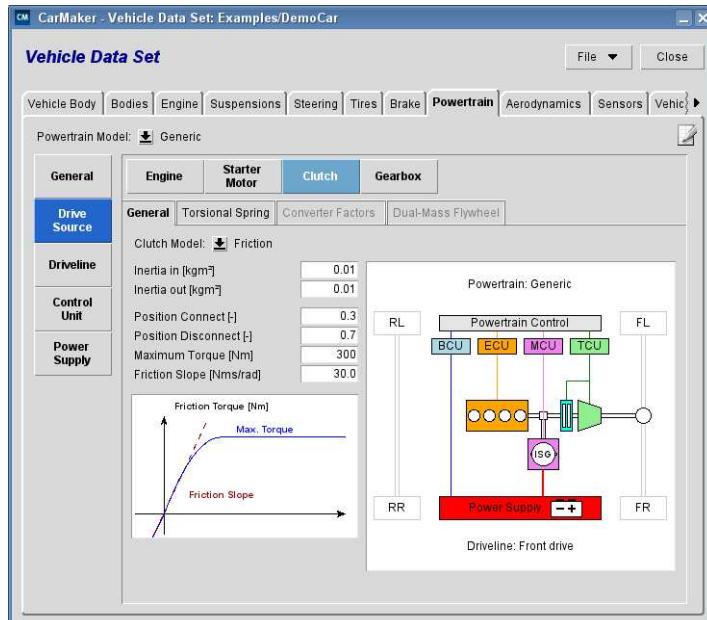


Figure 5.68: Clutch model friction parametrization

This model is suitable for all usual manual gear boxes found on most of European vehicles, for automated manual gear boxes, but not for double DCTs (Dual Clutch Transmission). The clutch is modeled by plates linked between each other with a virtual spring/damper system. It is opened by the maneuver control (e.g. IPGDriver) or the TCU depending on the gearbox model that it belongs to.

<b>Position Connect (-)</b>	At this clutch pedal position/signal from TCU the clutch starts slipping when opening. Values from 0 (pedal released) to 1 (pedal fully pressed).
<b>Position Disconnect (-)</b>	At this clutch pedal position/signal from TCU the clutch starts transferring torque when closing. Values from 0 (pedal released) to 1 (pedal fully pressed).
<b>Maximum Torque (Nm)</b>	This parameter sets the maximum transmissible torque which is equal for both cases, clutch is slipping and clutch is closed.
<b>Friction Slope (Nm/rad)</b>	Equivalent to the friction coefficient. It defines the characteristic of the origin of the friction torque curve (see <a href="#">Figure 5.69</a> ).

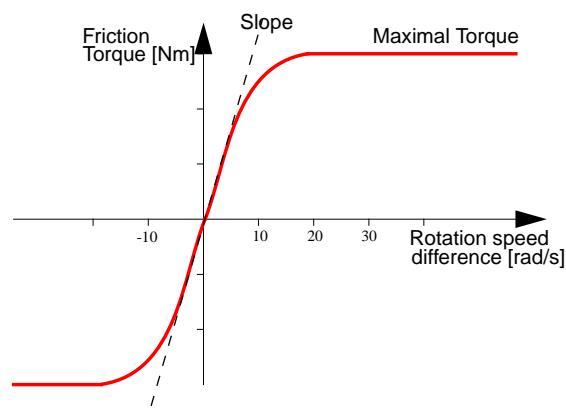


Figure 5.69: Friction torque curve

<b>Torsional Spring Mode</b>	The elasticity of the clutch is modeled by a torsional spring. There are two modes available to describe its stiffness:
------------------------------	---

Table 5.69: Torsional Spring Modes

Mode	Description
Characteristic Value	The spring stiffness corresponds to a constant value.
1D Look-Up Table	For different relative angles between the clutch input and output side different spring torques are defined.

<b>Torsional Spring Constant (Nm/rad)</b>	This parameter describes the clutch elasticity as a constant torsional spring stiffness for spring mode <i>Characteristic Value</i> only.
---	---

<b>Amplification (-)</b>	This factor is applied to the spring torques defined in the <i>1D Look-Up Table</i> . You can use this parameter to change the torsional spring characteristics quickly.  For more information about the clutch model <i>Friction</i> , please read the <a href="#">Reference Manual section 'Clutch Model Friction'</a> . Note that this model is a functional model, and not a physical model.
--------------------------	--

## Clutch Model: Converter

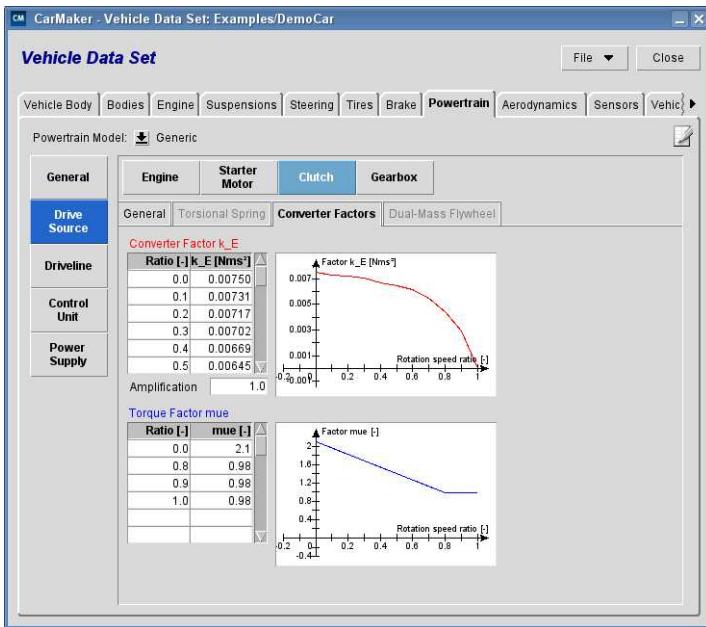


Figure 5.70: Clutch model Converter parametrization

The *Converter* model would usually be chosen in combination with an automatic gearbox, but not for CVTs. The model corresponds to a hydraulic torque converter also known as Föttinger converter. It includes a lock-up clutch that is supposed to mechanically connect both sides of the clutch in case their rotation speeds are close in order to increase the clutch efficiency.

- Converter Factor  $k_E$**  The input torque conversion factor  $k_E$  that is defined in this table characterizes the relation between the rotation speeds of the converter parts and the input torque. It includes the multiplication with the outer diameter. The factor  $k_E$  depends on the rotation speed ratio (*Ratio*) between output and input side. This table must contain values between 0 and 1. The part between 1 and 2 is mirrored.
- Amplification (-)** This parameter enables to scale the factor  $k_E$  very quickly instead of modifying the entire table itself. You can also use this parameter to convert the value you have for the factor in the units required by CarMaker.
- Torque factor  $\mu_e$  (-)** The factor  $\mu_e$  describes the relation between converter input and output torque ( $\mu_e = T_{out}/T_{in}$ ). It depends on the rotation speed ratio (*Ratio*) between output and input side.
- Spring Constant (Nm/rad)** This spring constant in the *General* tab defines the stiffness of the converter's lock-up clutch.
- Position Disconnect (-)** At this lockup position/signal from TCU the clutch starts transferring torque when closing. Values from 0 (closed) to 1 (open). This parameter is located in the tab *General*.
- For more information about the clutch model *Converter*, please read the [Reference Manual](#) section '[Clutch Model Converter](#)'.

## Clutch Model: Dual-Mass Flywheel

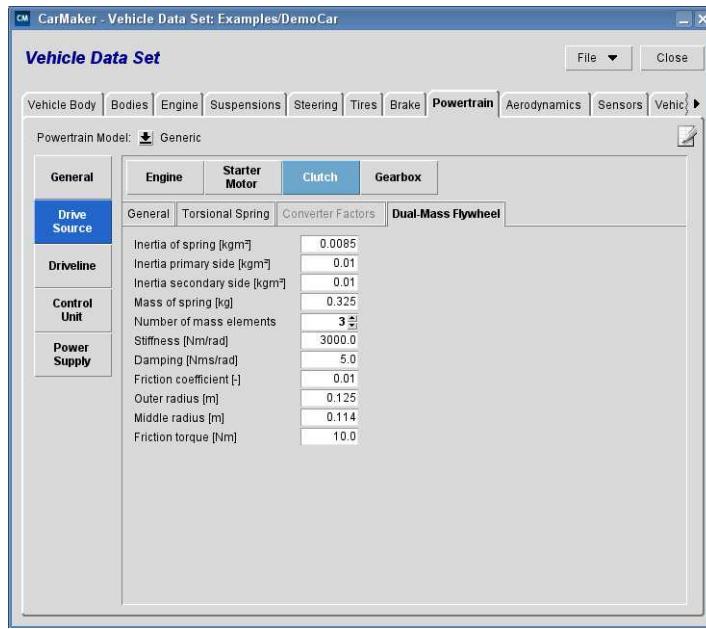


Figure 5.71: Clutch model Dual-Mass Flywheel parametrization

A dual-mass flywheel is used in order to reduce powertrain vibrations coming from the engine. In CarMaker's *Dual-Mass Flywheel* it is combined with a friction clutch model as described in [section 'Clutch Model: Friction'](#). Please refer to this section for all parameters of the friction clutch model. There are some additional parameters for the dual-mass flywheel:

<b>Inertia of spring (kgm<sup>2</sup>)</b>	Inertia of the dual-mass flywheel spring.
<b>Inertia of primary side (kgm<sup>2</sup>)</b>	Overall inertia of the elements that belong to the dual-mass flywheel primary side. The primary side is the side that is closer to the engine.
<b>Inertia of secondary side (kgm<sup>2</sup>)</b>	Overall inertia of the elements that belong to the dual-mass flywheel secondary side.
<b>Mass of spring (kg)</b>	Mass of the spring inside the dual-mass flywheel.
<b>Number of mass elements</b>	Number of mass elements into which the spring is divided. Value between 1 and 5.
<b>Stiffness (Nm/rad)</b>	Stiffness of the dual-mass flywheel spring element.
<b>Damping</b> <b>(Nms/rad)</b>	Damping of the dual-mass flywheel spring element.
<b>Friction coefficient (-)</b>	Friction coefficient at the contact between spring mass and primary side.
<b>Outer radius (m)</b>	Outer radius of the dual-mass flywheel spring element

**Middle radius (m)** Middle radius of the dual-mass flywheel spring element

**Friction torque (Nm)** Constant friction torque between the primary and secondary side.

For more information about the clutch model *Dual-Mass Flywheel*, please read the [Reference Manual section 'Clutch Model Dual-Mass Flywheel'](#).

### Clutch Model: DVA

The DVA clutch model is for very specific use. In this mode, the torque transmitted by the clutch can be controlled either directly by the user, or by an external model (see Programmer's Guide and [section 6.3 'DVA: Online Manipulation of the Simulation' on page 357](#) in this manual). To modify the transmitted torque, the User Accessible Quantity *PT.Clutch.DVA.Trq\_A2B* (in case of an internal gearbox clutch, replace *Clutch* by *<gearbox name>.clutch*) is used, the global clutch parameters described above are still available.

**DVA Clutch Type** Additionally you need to provide the information if the logic behind the torque transmission is closer to a friction clutch or a converter.

For more information about the clutch mode *DVA*, please read the [Reference Manual section 'Clutch Model DVA'](#). More information about the DVA mechanism itself can be found in this document, [section 6.3 on page 357](#).

### Clutch Model: External File

External File is for even more specific usage, e.g. if you developed your own clutch model (Power User level).

## 5.18.6 Parametrizing the Planetary Gear

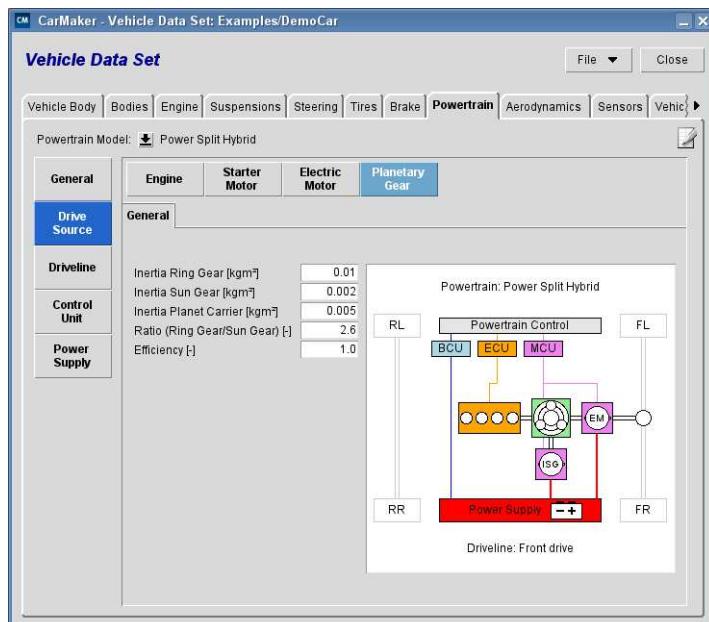


Figure 5.72: Planetary Gear parametrization

The planetary gear replaces gearbox and clutch the Drive Source of the powertrain model *Power Split Hybrid*. In CarMaker, the following configuration is available:

- The combustion engine is connected to the planet carrier

- The integrated starter generator is connected to the sun gear
- The electric motor is connected to the rig gear and driveline

The planetary gear is described by following parameters:

**Inertia Ring Gear (kgm<sup>2</sup>)** This parameter specifies the inertia of the ring gear.

**Inertia Sun Gear (kgm<sup>2</sup>)** This parameter specifies the inertia of the sun gear.

**Inertia Planet Carrier (kgm<sup>2</sup>)** This parameter specifies the inertia of the cage (planetary gear).

**Ratio (-)** The transmission ration defined here is the transmission ratio between ring gear and sun gear.

**Efficiency (-)** The efficiency defined here is the efficiency of torque conversion between ring gear and sun gear.

For more information about this model, please refer to [Reference Manual chapter 'Powertrain model Power Split Hybrid'](#).

## 5.19 Powertrain: Driveline

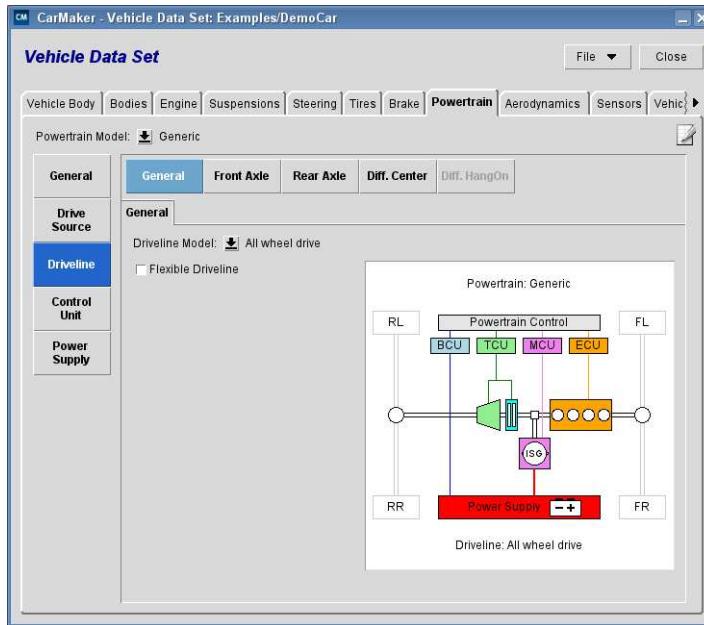


Figure 5.73: Selecting a driveline model

The Driveline models connect the Drive Sources to the wheels. The driveline models consist of shafts and differentials and realize different torque distributions.

**Driveline Model** You can choose one of the following driveline models.:

Table 5.70: Driveline models

Name of the transmission kind	Description
Front drive	Front wheel drive configuration for powertrain architectures with only one Drive Source
Rear drive	Rear wheel drive configuration for powertrain architectures with only one Drive Source
All wheel drive	Permanent all wheel drive configuration, with fixed torque ratio between the front and rear axles for powertrain architectures with only one Drive Source
Front drive, rear axle shiftable	Front wheel drive configuration with temporary torque transfer to the rear wheels if front wheels spin (variable torque distribution between the front and rear axles) for powertrain architectures with only one Drive Source
Rear drive, front axle shiftable	Rear wheel drive configuration with temporary torque transfer to the front wheels if rear wheels spin (variable torque distribution between the front and rear axles) for powertrain architectures with only one Drive Source
Universal drive	Driveline model for powertrain architectures with multiple Drive Sources; Drive Sources and wheels can be connected by differentials, by shafts or directly (e.g. wheel hub motors)

**Flexible Driveline** By choosing this option, every generic driveline model can be simulated with flexible half and drive shaft. In this case, each shaft has an additional degree of freedom for the shaft rotation. For additional information about flexible driveshafts, see [section 'Flexible Shafts' in the Reference Manual](#).

### Driveline Model: Front drive / Rear drive

The driveline models *Front drive* and *Rear drive* are quite similar. The only difference is that the output shafts of the differential are connected to the front left and right wheel for *Front drive*, but to the rear left and right wheel for *Rear drive*. That is why they are described together.

**External Torque off** If this option is selected, no external torques are applied to the non-driven axle / wheels.

**External Torque to Differential** In case of a front or rear driven vehicle there is only one differential placed on the driven axle. However, there is the possibility to apply an additional torque to the non-driven axle (e.g. coming from an electric motor). Using this option, the external torque can be applied to a second differential which is positioned on the non-driven axle. The torque is defined by the DVA quantity *PT.DL.<pos>Diff.Trq\_Ext2Diff*.

**External Torque to Wheels** If this option is selected, an external torque can be applied directly to the wheels of the non-driven axle (no additional differential). The torque is defined by the DVA quantity *PT.W<Pos>.Trq\_Ext2W*.

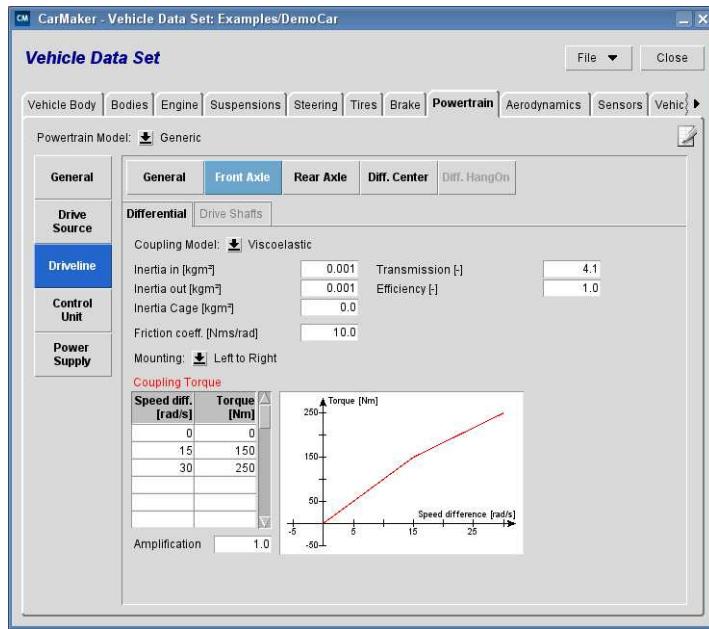


Figure 5.74: Differential Parametrization

The **Differential** model that is used here transfers the torque from the input shaft to each wheel, or for a mid-differential to each axle. On the other hand the speeds coming from both output shafts are converted to a mid speed imposed to the input shaft. The model is described by the following general parameters.

#### Coupling Model

The coupling models modify the torque distribution to the left and right wheel. Please refer to [section 5.19.1 'Parameterizing the Differential Coupling Model'](#) for a list of coupling models available and their parameters.

#### Inertia in ( $\text{kgm}^2$ )

This inertia includes all rotating and spinning components at the input side of the differential whose inertia has not been taken into account by another parameter. The components of the cage are not included in this parameter.

#### Inertia out ( $\text{kgm}^2$ )

 This inertia includes all rotating and spinning differential output shaft components. The components of the cage are not included in this parameter. The inertia of only one side of the driveline from the differential to the wheels should be parameterized. CarMaker automatically uses the same value for the other side.

There are different possibilities to attribute the inertias of the shafts connecting gearbox output and rear/front differential input to one of these two elements. In CarMaker, it is up to you to choose, which part of the inertias should be added to the gearbox output or differential input. In case of *All wheel* drive, the same applies for the shafts (and other components) between gearbox output and center differential input as well as center differential output and front/rear differential input.

#### Inertia Cage ( $\text{kgm}^2$ )

This inertia includes the inertia of the cage and of all rotating and spinning components that are in it.

#### Transmission (-)

This parameter is the transmission ratio from differential input shaft to cage. For front or rear differential, the transmission  $I$  is defined as  $I = (T_{\text{out left}} + T_{\text{out right}})/T_{\text{in}}$ ,  $T_{\text{out left}}$  and  $T_{\text{out right}}$  being the output torque on the left and right side (after the differential), and  $T_{\text{in}}$  the input torque (ahead of the differential).

#### Efficiency (-)

This parameter sets the state efficiency of the differential. Values must be between 0...1.

If **Flexible Driveline** is selected, the **Drive Shafts** need to be parameterized, too.

**Flexible Half Shaft Stiffness (Nm/rad)**

This parameter only is available for the front and rear differential with activated flexible driveline. Defines the stiffness coefficient of the left and right flexible half shaft.

**Flexible Half Shaft Damping (Nms/rad)**

This parameter only is available for the front and rear differential with activated flexible driveline. Defines the damping coefficient of the left and right flexible half shaft.

**Flexible Half Shaft Inertia (kgm<sup>2</sup>)**

This parameter only is available for the front and rear differential with activated flexible driveline. Defines the inertia of the left and right flexible half shaft.

For more information about the differentials, please read the [Reference Manual section 'DriveLine - Generic'](#).

## Driveline Model: All wheel drive

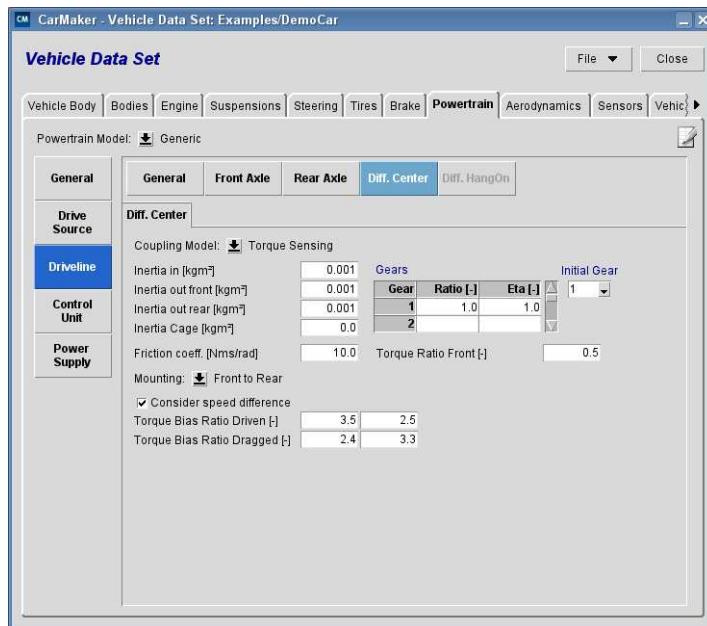


Figure 5.75: Center Differential parametrization

The *All wheel drive* model consists of three differentials. The center differential's input torque comes from the gearbox output and is distributed to a front and a rear differential. The parameters for each differential are the same as for *Front drive* and *Rear drive* (see [section 'Driveline Model: Front drive / Rear drive'](#)) except the following parameters.

**Inertia out front (kgm<sup>2</sup>)**

This inertia includes all rotating and spinning differential output shaft components that connect center differential and front axle differential. The components of the cage are not included in this parameter.

**Inertia out rear (kgm<sup>2</sup>)**

This inertia includes all rotating and spinning differential output shaft components that connect center differential and rear axle differential. The components of the cage are not included in this parameter.

**Gears**

In CarMaker, it is possible to simulate a shiftable center differential. In the table **Gears** you can set up to four different gears with their transmission ratio (*Ratio*) and their efficiency (*Eta*). The transmission ration *I* is defined as  $I = (T_{\text{out front}} + T_{\text{out rear}}) / T_{\text{in}}$ .

$T_{\text{out front}}$  and  $T_{\text{out rear}}$  being the output torque on the front and rear (after the center differential), and  $T_{\text{in}}$  the input torque (ahead of the differential).

In case of a non-shiftable differential, you only need to parameterize the first gear.

<b>Initial Gear</b>	If a shiftable differential is defined (more than one differential gear), you can select the initial differential gear that is used by default. The differential gear can be switched via Direct Variable Access to the quantity <i>PT.Gen.DL.CDiff.GearNo</i> .
<b>Torque Ration Front (-)</b>	This parameter specifies the percentage of the input torque which is transferred to the front axle (the remaining torque goes to the rear axle).  If <b>Flexible Driveline</b> is selected, the <b>Drive Shafts</b> need to be parameterized, too.
<b>Flexible Drive Shaft Stiffness (Nm/rad)</b>	This parameter only is available for the front and rear differential with activated flexible driveline. Defines the stiffness coefficient of the flexible drive shaft between center differential and front/rear differential.
<b>Flexible Drive Shaft Damping (Nms/rad)</b>	This parameter only is available for the front and rear differential with activated flexible driveline. Defines the damping coefficient of the flexible drive shaft between center differential and front/rear differential.  For more information about the differentials, please read the <a href="#">Reference Manual section 'DriveLine - Generic'</a> .
<b>Driveline Model: Front drive, rear axle shiftable / Rear drive, front axle shiftable</b>	
These driveline models are quite similar to <i>Front drive</i> and <i>Rear drive</i> , but they contain an additional hang on differential. A hang on differential corresponds to clutches used mainly in SUVs that transfer a part of the torque to the other axle if the wheels start to spin. It is in fact a plain front or rear wheel drive configuration, which becomes an all wheel drive if the vehicle loses grip.	
The parameters for each differential are the same as for <i>Front drive</i> and <i>Rear drive</i> (see section <a href="#">'Driveline Model: Front drive / Rear drive'</a> ) except the following parameters.	
<b>Disconnect (rpm)</b>	This parameter is only used by hang on differentials. It defines from which shaft rotation speed the transmission is suspended if the vehicle is dragged, e.g. from this speed no more torque is transferred to the rear axle, even if the wheels of the front axle slip or vice versa.
<b>Diff. HangOn Transmission (-)</b>	The transmission ratio of the hang on differential is defined as $I = (T_{\text{out front}} + T_{\text{out rear}}) / T_{\text{in}}$ $T_{\text{out front}}$ and $T_{\text{out rear}}$ being the output torque to the front and rear shafts (after the differential), and $T_{\text{in}}$ the input torque (ahead of the differential).   The overall transmission ratio between the engine speed and the wheel speed has to be the same for both front and rear axles! Pay attention to set the transmission ratio of all of the differentials correctly.  For more information about the differentials, please read the <a href="#">Reference Manual section 'DriveLine - Generic'</a> .

## Driveline Model: Universal drive

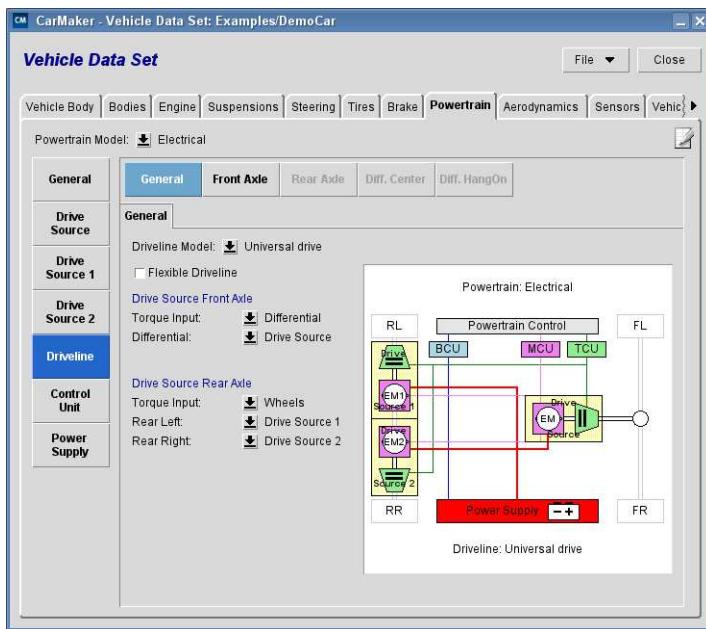


Figure 5.76: Universal Drive parametrization

If your powertrain architecture consists of several Drive Sources, you should use the *Universal* drive line model.

**Torque Input** It enables you to connect the Drive Sources output shafts to the wheels of each axle in four different ways:

Table 5.71: Universal Drive torque input

Model	Description
Differential	One drive source is connected to the axle via a differential (see section 'Driveline Model: Front drive / Rear drive' for differential parameters). The driving torque is supported by the vehicle body. You can choose the drive source that should be connected to the differentials input under "Differential". The differential's output shafts are connected to the left and right wheel of the axle.
Driveshafts	One drive source is connected to one wheel of this axle via a simple shaft. The driving torque is supported by the vehicle body. You can select the drive sources connected to the left and right wheel under "Front Left"/"Rear Left" and "Front Right"/"Rear Right".
Wheels	One drive source is connected directly to one wheel of this axle. The driving torque is supported by the wheel carrier. This is the typical configuration for wheel hub motors. You can select the drive sources connected to the left and right wheel under "Front Left"/"Rear Left" and "Front Right"/"Rear Right"
not specified	No torque input to this axle



It is possible to apply external torques at a wheel or differential. This enables you to set up new powertrain topologies by modelling only parts (additional Drive Sources). The driveline model *Universal* differentiates two cases:

- If there already is a Drive Source connected at this torque input, the external torque can be set via DVA on the User Accessible Quantity  $PT.DL.DriveSrc.<ds>.Trq\_ext$  where  $<ds>$  stands for the Drive Source at this torque input.
- If there is not yet a Drive Source at this torque input, it is possible to write an external torque to the User Accessible Quantity  $PT.DL.DriveSrc.<ds>.Trq\_in$ . Here  $<ds>$  stands for a Drive Source that does not exist in the powertrain model yet. E.g. if the powertrain model selected in the Vehicle Data Set is a *Generic* powertrain (that means there is just one Drive Source 0), Drive Source 1, 2 or 3 can be used for an external torque. This external torque is linked to a wheel or differential by selecting the Drive Source  $<ds>$  as torque input.

## 5.19.1 Parameterizing the Differential Coupling Model

For each differential of the driveline, you have the possibility to select a coupling model that influences the torque distribution to the differentials output shaft

<b>Coupling Model</b>	There are up to seven different models available that describe how both output sides of the differential are linked together:
-----------------------	---

Table 5.72: Coupling Models for the Differential

Coupling Model	Description	Differential Position
not specified	No coupling between both output shafts. In this case no differential locking system is selected.	front / rear, center, hanged on
Viscoelastic	Differential locking system with hydromatic converter. As soon as there is a rotating speed difference between both output shafts, a part of the torque is transferred from the shaft which rotates faster to the shaft which rotates slower, according to a characteristic parameterization under <i>Coupling Torque</i> .	front / rear, center, hanged on
Torque Sensing	Mechanical differential locking system. With this coupling model, a part of the torque can be oriented up to a parameterized limit to the shaft that has more grip (lower rotating speed). The torque transfer only occurs if a wheel is slipping.	front / rear, center,
Torque Vectoring	The torque can be distributed freely to the output shafts via modification of DVA quantities or directly in the C code level. This option makes sense only together with a Simulink or with a C-code model which defines the torque distribution strategy.	front/rear, center
Locked	This coupling model acts like a rigid connection between input and output shaft.	front/rear, center, hanged on
Locked by DVA	Differential locking system based on the <i>Torque Sensing</i> coupling model. Instead of being fixed, the locking torque can be altered via modification of the DVA quantities.	front/rear, center, hanged on
External File	Used to load a clutch parameterization file corresponding to an user clutch model.	front/rear, center, hanged on



Some advises to choose the correct differential model:

- If you have a plain differential without any locking system, choose the option *not specified*
- If your vehicle is equipped with a viscous system (e.g. system "Haldex"), choose the option *Viscoelastic*.
- If you wish to model a vehicle equipped with a mechanical system (e.g. system "Torsen"), choose the option *Torque Sensing*.
- If you want to simulate a locked differential, choose the option *Locked*.
- For the implementation of your own model, either in Simulink or in the C-level, choose either the option *Locked by DVA* or *Torque Vectoring*. The DVA mode enables you to tune online the locking torque that acts only when a wheel is slipping. *Torque Vectoring* enables to give freely at any moment a part of the torque to one or another output shaft.
- If your differential's behavior does not fit any of the pre-implemented coupling models, you also have the possibility to add your own model to your CarMaker project.

## Coupling Model: Viscoelastic

<b>Mounting</b>	The mounting position defines which differential shafts are coupled and thus are affected by the torque transfer defined by the table <i>Coupling Torque</i> . Possible values for front / rear differentials: Left to Right, Input to Left, Input to Right. Possible values for a center differential: Front to Rear, Input to Front, Input to Rear
-----------------	--

<b>Coupling Torque</b>	This table defines the torque transition. According to the output shaft rotating speed difference ( <i>speed diff.</i> ), the differential will reduce the torque to the shaft which rotates faster and increase the torque to the shaft which rotates slower by the value stated in the column <i>torque</i> .
------------------------	---

<b>Amplification (-)</b>	This parameter enables to scale the torque characteristics parameterized in the table very quickly instead of modifying the entire table itself. You can also use this parameter to convert the value you have for the factor in the units required by CarMaker. Usually, leave the standard value to 1.  Additional information about the coupling torque characteristics can be found in the <a href="#">Reference Manual</a> , section ' <i>Viscoelastic</i> '
--------------------------	---

## Coupling Model: Torque Sensing

When a wheel, respectively a shaft, is slipping, the differential is able to transfer a part of the torque to the wheel or shaft without slip, within a given limit. To parameterize this behavior, the *Torque Bias Ratios (TB) Driven* and *Dragged* are used.

<b>Friction coeff. (Nms/rad)</b>	With the locking system, you consider two states: when there is no shaft speed difference (state "stick"), and when there is a shaft speed difference (state "slip"). In the latter case, a differential locking system enables to give more torque to the shaft which has the lower rotating speed, i.e the wheels that have more grip. The change from the stick to the slip state is modeled with a damper / spring element. This leads to vibrations in the driveline. With the parameter Friction coeff. you can parameterize the damping behavior that is the most relevant part of this model and cushion those vibrations.
----------------------------------	---

<b>Consider speed difference</b>	If this option is selected, the torque bias ratio changes depending on the output shaft that rotates faster.
----------------------------------	--

<b>Torque Bias Ratio Driven/Dragged (-)</b>	<p>The meaning of this parameters becomes clear with a little example. For example: Let us suppose that the torque bias is set to 2, then the differential will limit the torque to the slipping wheel to 33.3% of the input torque and give 66.6% to the other wheel (twice as much as the spinning wheel).</p> <p>When <i>Consider speed difference</i> is selected, two torque bias ratios can be set for <i>driven</i> and <i>dragged</i>. The first value applies if the left respectively the front shaft speed is bigger than the right respectively rear shaft speed. Otherwise the second value is used.</p> <p><i>Driven</i> means that this parameter applies to the torque coming from the input shaft (positive torque).</p> <p><i>Dragged</i> means that this parameter applies to the torque that goes to the input shaft (negative torque, when decelerating).</p> <p>Additional information about the torque bias definition can be found in the <a href="#">Reference Manual</a>, section '<a href="#">Torque Sensing</a>'.</p>
---	---

## Coupling Model: Torque Vectoring

<b>Friction coeff. (Nms/rad)</b>	<p>With the locking system, you consider two states: when there is no shaft speed difference (state "stick"), and when there is a shaft speed difference (state "slip"). In the latter case, a differential locking system enables to give more torque to the shaft which has the lower rotating speed, i.e the wheels that have more grip.</p> <p>The change from the stick to the slip state is modeled with a damper / spring element. This leads to vibrations in the driveline. With the parameter Friction coeff. you can parameterize the damping behavior that is the most relevant part of this model and cushion those vibrations.</p>
--------------------------------------	--

<b>Torque Ratio Left (-)</b>	<p>This model is characterized by the single parameter <i>Torque Ratio Left</i>. It defines the initial torque distribution of the differential between both output shafts. For a center differential, this data is already defined by the parameter <i>Torque Ratio Front</i>.</p> <p>This value can be changed during the simulation via modification of DVA quantities <i>PT.Gen.DL.FDiff.TrqRatio</i> or <i>PT.Gen.DL.RDiff.TrqRatio</i>.</p>
----------------------------------	---

## Coupling Model: Locked

This coupling acts like a rigid connection between input and output shaft. it has no special parameters.

## Coupling Model: Locked by DVA

Using this coupling, you can lock the differential by writing a locking torque to the quantity *PT.Gen.DL.<pos>Diff.DVA.Trq\_A2B* (replace *<pos>* by the position of the differential: *F* for front, *R* for rear, *C* for center). The shaft A is either the left shaft (for a front or rear differential), or the front shaft (for a mid or hang on differential).

This option should be used only if you want to implement your own differential locking system, for instance by a Simulink model. For the differential to always be locked, provide a large locking torque by DVA (e.g. 10000 Nm).

## Coupling Model: External File

External File is for even more specific usage, e.g. if you developed your own coupling model (Power User level).

## 5.20 Powertrain: Control Units

With CarMaker 5.0, a new level has been introduced inside the powertrain model: The Control Units. These models correspond to the software part of the powertrain elements. Their task is to provide control signals (e.g. motor load) to the electromechanical powertrain components and to calculate/ "measure" characteristic values that describe the current state of these components (e.g. battery state of charge). Four control units are attributed to the different types of electromechanical powertrain components:

- ECU (Engine Control Unit) - combustion engine
- TCU (Transmission Control Unit) - gearboxes and clutches
- MCU (Motor Control Unit) - electric motors including the integrated starter motor
- BCU (Battery Control Unit) - power supply including batteries

In contrast, PT Control (PowerTrain Control) is a global control unit whose main task consists in providing target values for the controller algorithms of the other control units. E.g. it defines target torques for the electric motors that are then transformed into a load signal by the MCU in order to converge the current motor torque to the target torque.

### 5.20.1 Parametrizing PT Control

PT Control (also known as hybrid control unit HCU in hybrid vehicles) is the highest order control unit. Its main tasks are

- Regulation of the vehicle operation states (off, accessories on, drive...) that the driver selects by turning the vehicle key and/or pushing the start-stop-button
- Mechanical energy management: Translating the driver's actions (e.g. gas pedal actuation) into targets for the electro mechanic components (e.g. target torque for the engine, target gears for the gearboxes)
- Electric energy management: Regulating the batteries states of charge and the energy transfer between different electric circuits
- Providing informations about maximum and current regenerative braking torque to the Brake Control Unit (compare [section 5.16.1 'Brake Control \(Hydraulic System\)'](#)) in order to organize energy recuperation

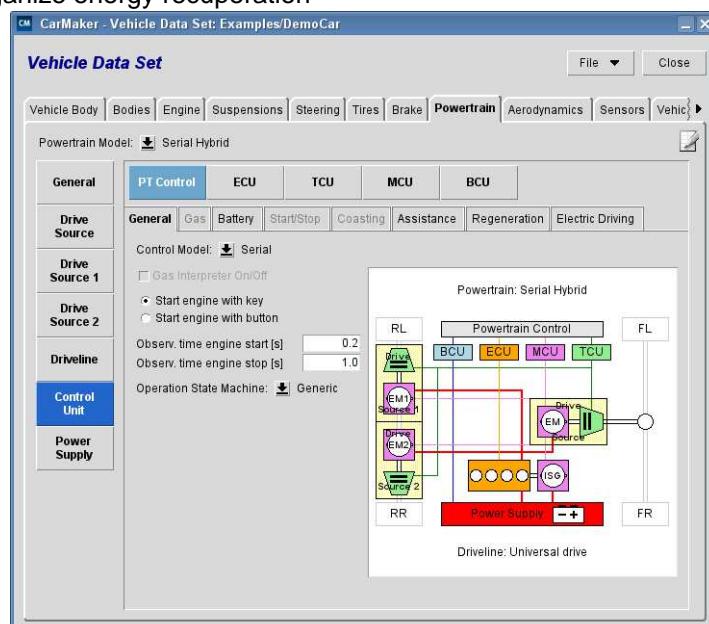


Figure 5.77: Selecting a PTControl model

**Control Model** The global control strategy contained in a PT Control model is closely linked to the powertrain architecture. Take care that the PT Control model that you select fits to your powertrain architecture. E.g. trying to drive a Generic powertrain with a PT Control implemented for Serial Hybrid would lead to a nonsensical vehicle behavior. CarMaker contains an example control strategy for each powertrain architecture:

Table 5.73: PT Control models

Model	Description
Generic	Conventional control strategy for <i>Generic</i> powertrain
Micro	Micro hybrid control strategy (with start-stop functionality) for <i>Generic</i> powertrain
Parallel P1	Mild hybrid control strategy for <i>Parallel Hybrid P1</i> powertrain
Parallel P2	Full Hybrid control strategy for <i>Parallel Hybrid P2</i> powertrain
Axle Split	Full Hybrid control strategy for <i>Axle Split Hybrid</i> powertrain
Power Split	Full Hybrid control strategy for <i>Power Split Hybrid</i> powertrain
Serial	Full Hybrid control strategy for <i>Serial Hybrid</i> powertrain (Range Extender)
Electrical	Control strategy for <i>Electrical</i> powertrain



There are a lot of different control strategies for each powertrain architecture, some are deterministic and based on well defined states that are switched under predefined conditions, others use fuzzy-logic or learning controls... and this is just an extract of all the existing approaches. The control models in [Table 5.73: PT Control models](#) are based on a global deterministic control strategy developed by IPG that includes the most common strategy modes (states) currently used for hybrid cars. [section 'IPG Powertrain Control Strategy' on page 259](#) as well as [Reference Manual chapter 'PT Control'](#) provide more information about this control strategy.

If the main focus of your work is not the hybrid control strategy (most of all, if you want to simulate conventional vehicles with only one combustion engine and no electric motors), you probably will be fine with using the IPG control strategies and maybe adapting some of their parameters to your vehicle. Otherwise you also have the possibility to replace the whole control strategy by your own model.

## IPG Powertrain Control Strategy

All PT Control models listed in [Table 5.73: PT Control models](#) are based on the same global control strategy that is described in this chapter. [Figure 5.78](#) shows its structure.

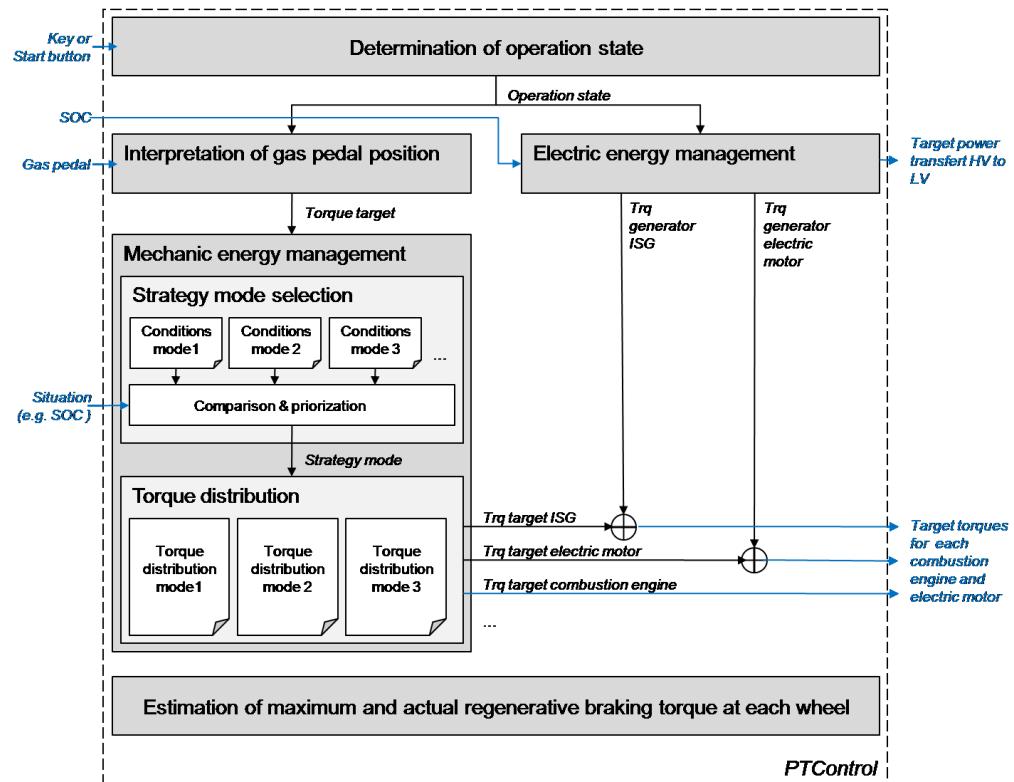


Figure 5.78: IPG Powertrain Control Strategy

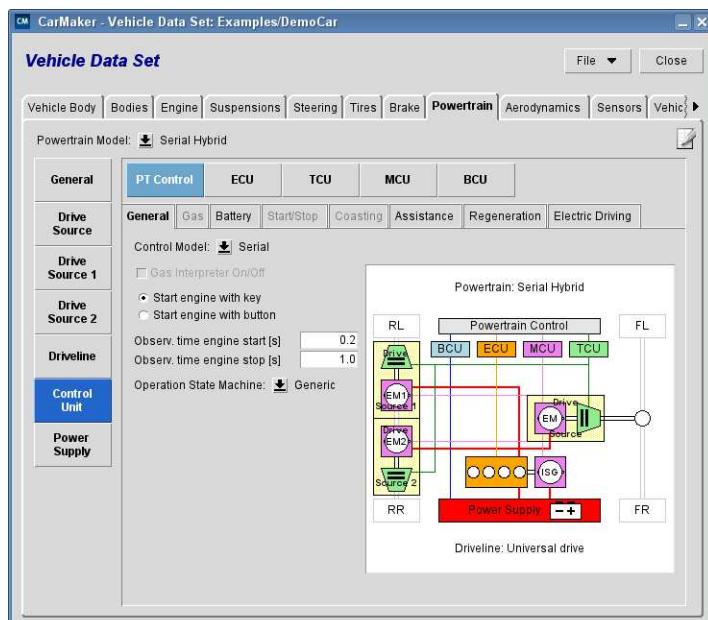


Figure 5.79: PT Control General parametrization

Image you are sitting in a real car and you want to start driving. Therefor you need to turn the key in the ignition lock until the engine is running. Usually there is at least one intermediate position where only some accessories (e.g. the radio) are activated, but the vehicle is not (yet) ready for driving. The task "Determination of Operation states" represents what is happening on vehicles side when you do this. The parameters for this submodels are set in the tab **General**.

**Start engine with key/button** This parameter describes the interface between the driver (Driving Manager) and the *Operation State Machine* described below. If *Start engine with key* is selected, the Operation State Machine expects the driving manager to turn a key in order to switch to the driving (or any other) state. If *Start engine with button* is selected, it waits for the vehicle start-stop-button to be pushed instead.

**Operation State Machine** You have the possibility to select two different Operation State Machines:

Table 5.74: Operation state machines

Model	Description
Generic	Generic Operation State Machine model that can be activated by a key or the start-stop-button
External File	Used to load a parameterization file corresponding to an user Operation State Machine

### Operation State Machine: Generic

This model differentiates six vehicle states:

- Absent: driver is outside the vehicle, power off
- PowerOff: driver is inside the vehicle, power off
- PowerAccessory: Electric power for accessories on
- PowerOn: electric power on (vehicle ignition)
- Start (only with key, conventional vehicle): Engine start
- Driving: Ready for driving or driving

More Information about this model can be found in Reference Manual section '[PTControlIOSM Model Generic](#)'.

### Operation State Machine: External File

External File is for even more specific usage, e.g. if you developed your own operation state machine model (Power User level).

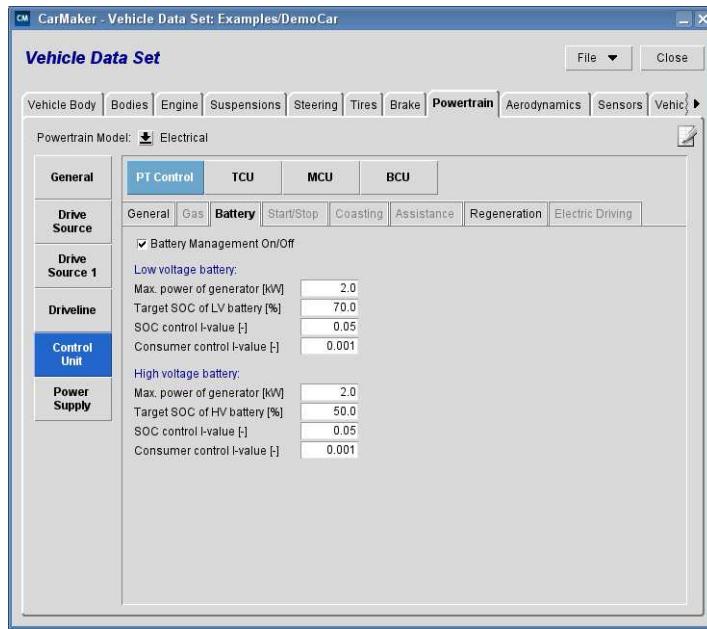


Figure 5.80: PT Control Battery parametrize

The electric energy management in IPG powertrain control strategy (compare to [Figure 5.78](#)) controls the batteries states of charge by assigning additional generator torques to the integrated starter motor and manages the energy transfer between different electric circuits. Its parameters can be set in the tab **Battery**.

<b>Battery Management On/Off</b>	You have the possibility to deactivate the battery management completely by deselecting this option. If it is activated, the following parameters are available for low and high voltage circuit.
<b>Max. power of generator (kW)</b>	This parameter corresponds to the maximum admissible generator power for recharging the battery in low/high voltage circuit (absolute value).
<b>Target SOC of LV/HV battery (%)</b>	This target state of charge of the low/high voltage battery is used as target value for the controller algorithm of the battery management.
<b>SOC control I-value (-)</b>	This control parameter scales the influence of the difference between target and current battery state of charge (integral element) on the additional generator torque demand.
<b>Consumer control I-value (-)</b>	This control parameter scales the influence of the auxiliary power consumption (integral element) on the additional generator torque demand.  The controller algorithms that are used on the battery management are described in <a href="#">Reference Manual chapter 'Battery management'</a> .

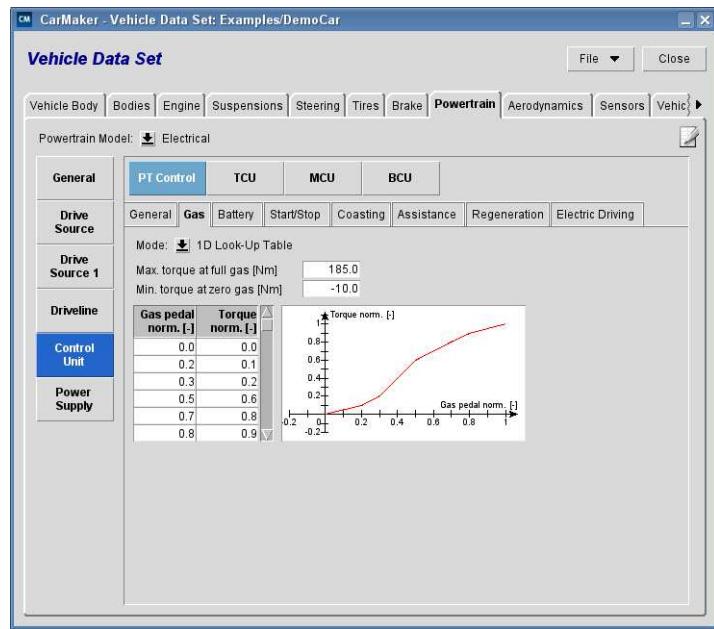


Figure 5.81: PT Control Gas parametrize

As in a real car, the driver of the virtual car communicates his/her will to accelerate or decelerate by pushing or releasing the gears pedal. IPG powertrain control model interprets the gas pedal position as an overall torque demand that is then distributed to the available engine and electric motors (see [Figure 5.78](#)). You can define this interpretation of the gas pedal position in the tab **Gas**



The PTControl model *Generic* includes two modes: Either it forwards the gas pedal position directly as a load signal to the MCU or it calculates a target torque based on gas pedal position and transfers this torque to the MCU. Select **Gas Interpreter On/Off** to use the second mode.

**Mode** The gas pedal position can be interpreted as a desired driving torque by three ways:

Table 5.75: Gas pedal interpretation modes

Mode	Description
<b>Characteristic Value</b>	Linear relation between gas pedal position and torque demand
<b>1D Look-Up Table</b>	Non-linear relation between gas pedal position and torque demand
<b>DVA</b>	Torque demand is defined by

#### Mode: Characteristic Value

This mode defines a linear relation between gas pedal position (*gas*) and torque demand ( $T_{target}$ ):  $T_{target} = (T_{max} - T_{min}) * gas + T_{min}$

**Max. torque at full gas (Nm)** This torque corresponds to the torque demand if the gas pedal is fully pressed ( $T_{max}$ ).

**Min. torque at zero gas (Nm)** This torque corresponds to the torque demand if gas pedal is not pressed at all ( $T_{min}$ ). It usually is negative.

**Mode: 1D Look-Up Table**

The look up table defines the normalized torque demand (*Torque norm.*,  $k$ ) depending on the gas pedal position (*Gas pedal norm.*,  $gas$ ). That lead to the torque demand  $T_{target} = T_{min} + k^*(T_{max} - T_{min})$

**Max. torque at full gas (Nm)** This torque corresponds to the torque demand if the gas pedal is fully pressed ( $T_{max}$ ).

**Min. torque at zero gas (Nm)** This torque corresponds to the torque demand if gas pedal is not pressed at all ( $T_{min}$ ). It usually is negative.

**Mode: DVA**

The DVA mode is for very specific use. In this mode, the total torque demand can be controlled either directly by the user, or by an external model (see Programmer's Guide and [section 6.3 'DVA: Online Manipulation of the Simulation' on page 357](#) in this manual). It does not even have to depend on the gas pedal position. To modify the torque demand, the User Accessible Quantity *PT.Control.GasInterpret.Trq\_trg* is used.

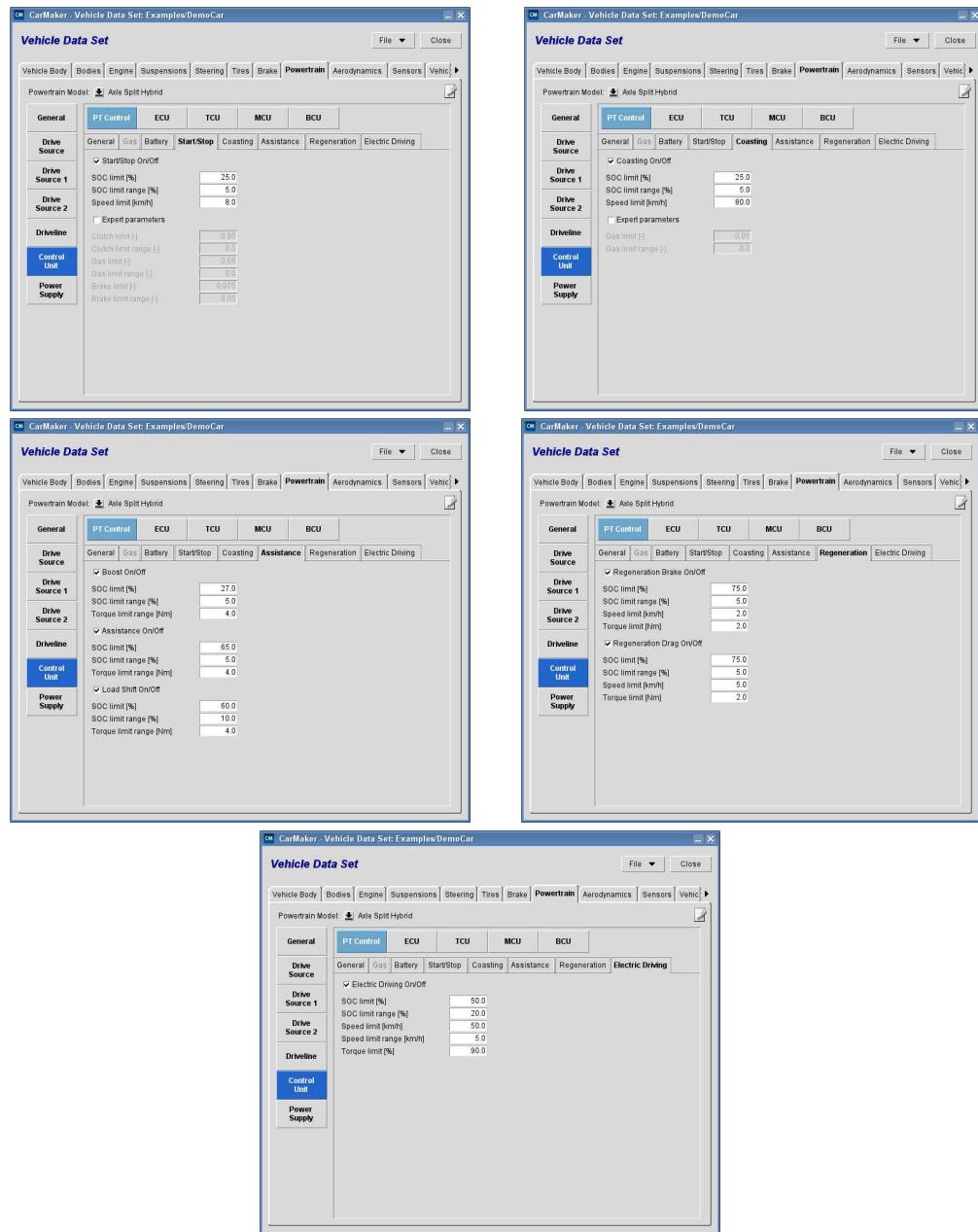


Figure 5.82: PT Control strategy modes parametrize

In the IPG powertrain control model, the mechanical energy management takes the torque demand that has been calculated by the gas pedal interpretation and distributes it to the engine and/or electric motor(s). This distribution is prescribed by strategy modes. CarMaker differentiates nine strategy modes:

Table 5.76: IPG powertrain control strategy modes

No	Mode	Description
0	Start/Stop	Standstill, engine off
1	Regenerative brake	Electric motor is used as a generator to apply a brake torque and thus recharge the battery
2	Regenerative drag torque	Electric motor is used as a generator to apply an artificial drag torque (if gas pedal is released)) and thus recharge the battery

Table 5.76: IPG powertrain control strategy modes

No	Mode	Description
3	Coasting	Engine off and decoupled, electric motors off
4	Electric drive	Driving with electric motors only, engine off
5	Load shift	Combustion engine runs at optimum torque while electric motor (as generator) regenerates the part of energy that is not needed for driving
6	Assist	Combustion engine runs at optimum torque while electric motor (as motor) applies the difference to the demanded torque
7	Boost	Combustion engine runs at maximum torque while electric motor applies the difference to the demanded torque
8	Engine drive	Driving with combustion engine only, electric motors off



For PT Control model *Serial*, the definition of some strategy modes is slightly different because there is no mechanical connection between combustion engine and wheels. Please refer to [Reference Manual chapter 'Powertrain model Serial Hybrid'](#) for this special case.

Conditions on the vehicle's state and driver input define which strategy mode currently is applied and when it switches to an other mode. If the conditions for several strategy modes are satisfied at the same time, there is a prioritization between the modes. Generally recuperation of energy is preferred to a neutral balance of energy and electric driving is preferred to driving with combustion engine. The mode numbers shown in [Table 5.76: IPG powertrain control strategy modes](#) corresponds to the priorities (0 = highest priority, 8 = lowest priority). During the transitions from modes with and without combustion engine running, there are intermediary states. This transition can be parameterized in the tab **General**, too.

**Observe. time engine start (s)** Before the engine is switched on, the demand to switch from a strategy mode, in which the engine is off, to a strategy mode, in which the engine is on, is observed. If it persists for the time defined here, engine is switched on.

**Observe. time engine stop (s)** Before the engine is switched off, the demand to switch from a strategy mode, in which the engine is on, to a strategy mode, in which the engine is off, is observed. If it persists for the time defined here, engine is switched off.

Of course, not all of the strategy modes are available for all powertrain architecture's control strategies. E.g. a conventional vehicle (PT control model *Generic*) only is able to drive with combustion engine whereas a full hybrid vehicle (e.g. PT Control model *Parallel P2*) can use all strategy modes:

Table 5.77: Available strategy modes

Strategy modes --> PTControl Model	0	1	2	3	4	5	6	7	8
Generic									x
Micro	x							x	x
Parallel P1	x	x				x	x	x	x
Parallel P2	x	x	x	x	x	x	x	x	x
Axle Split	x	x	x	x	x	x	x	x	x
Power Split	x	x	x	x	x	x	x	x	x

Table 5.77: Available strategy modes

Strategy modes --> PTControl Model	0	1	2	3	4	5	6	7	8
Serial		x	x		x	x	x		
Electrical		x	x		x				

For each mode, you have the possibility to set some parameters of the conditions to switch into this strategy mode and to stay in this strategy mode. In this document, we will only give a short overview. Please refer to [Reference Manual chapter 'Mechanical energy management - Strategy modes'](#) for a detailed description.

<b>&lt;Mode&gt; On/Off</b>	If you unselect this checkbox, the strategy mode <i>&lt;mode&gt;</i> is not used by the control strategy.
<b>SOC limit (%)</b>	State of charge of the main battery (usually high voltage battery) that needs to be passed to switch into this strategy mode. Depending on the strategy mode it refers to, this can be an upper (e.g. for mode electric drive) or lower (e.g. for mode regenerative braking) limit.
<b>SOC limit range (%)</b>	<p>For more stability, the SOC limit that needs to be passed to enter a mode is not exactly the same as the SOC limit that needs to be passed to stay in this mode once it is active. This range defines a zone around the <i>SOC limit</i>. This means e.g. for mode electric driving:</p> <ul style="list-style-type: none"> <li>Condition to enter mode electric drive <math>SOC &gt; SOC\ limit + 0.5 * SOC\ limit\ range</math></li> <li>Condition to stay in mode electric drive <math>SOC &gt; SOC\ limit - 0.5 * SOC\ limit\ range</math></li> </ul> <p>Please be aware that this is only a part of the condition. There might be other conditions (e.g. on vehicle speed) that have to be fulfilled or other modes with higher priority that can become active.</p>
<b>Speed limit (km/h)</b>	Vehicle speed that needs to be passed to switch into this strategy mode. This is always an upper limit.
<b>Speed limit range (km/h)</b>	<p>For more stability, the speed limit that needs to be passed to enter some modes is not exactly the same as the speed limit that needs to be passed to stay in the same mode once it is active. This range defines a zone around the <i>Speed limit</i>. This means e.g. for mode electric driving:</p> <ul style="list-style-type: none"> <li>Condition to enter mode electric drive <math>vehicle\ speed &lt; Speed\ limit - 0.5 * Speed\ limit\ range</math></li> <li>Condition to stay in mode electric drive <math>vehicle\ speed &lt; Speed\ limit + 0.5 * Speed\ limit\ range</math></li> </ul> <p>Please be aware that this is only a part of the condition. There might be other conditions (e.g. on SOC) that have to be fulfilled or other modes with higher priority that can become active.</p>
<b>Torque limit (Nm)</b>	For the <i>Regeneration</i> modes, the torque demand (absolute value) has to exceed a limit to activate regenerative braking or drag.
<b>Torque limit range (Nm)</b>	The <i>Assistance</i> modes become active and stay active if the torque demand passes the maximum and/or optimum combustion engine torque. You can define a torque range to this limit to differentiate between the condition to enter the mode and the condition to stay inside the mode. This stabilizes the control strategy by avoiding frequent changes between two modes ("oscillations").

## 5.20.2 Parametrizing the ECU

The engine control unit (ECU) contains the controllers of the combustion engine, especially the idle speed controller and a load controller. The latter provides a load signal for the engine based on the target rotation speed or target torque provided by PT Control and the engines current rotation speed or torque.

- ECU Model** Two ECU models are available in CarMaker:

Table 5.78: ECU Models

Mode	Description
Basic	Basic ECU model with PI controllers for load control
not specified	No ECU model (e.g. for electrical powertrain that does not contain any combustion engine)

### ECU Model: Basic

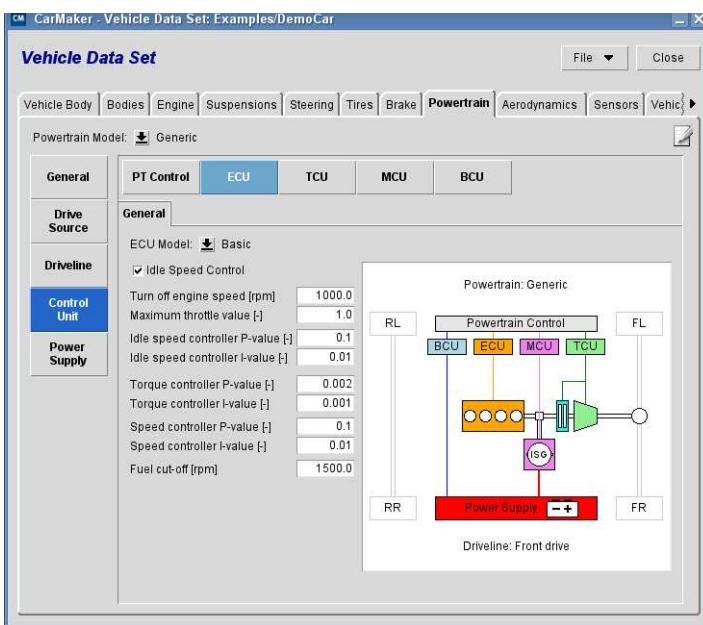


Figure 5.83: ECU model basic

**Idle Speed Control** Deselecting this box deactivates the ECU's idle speed controller.

**Turn off engine speed (rpm)** If the engine rotation speed exceeds the limit defined here, the idle speed controller is turned off. The turn off engine speed usually is about the engine idle speed + 200rpm.

**Maximum throttle value (-)** This value limits the engine load signal (throttle position) at the idle speed controller's output. Its value is between 0 and 1.

**Idle Speed controller P/I -value (-)** These parameters correspond to the proportional (*P-value*) and integral (*I-value*) of the idle speed controller which mainly is a PI-controller.

<b>Torque controller P/I -value (-)</b>	These parameters correspond to the proportional ( <i>P-value</i> ) and integral ( <i>I-value</i> ) of the torque speed controller which mainly is a PI-controller. The torque controller becomes active if the powertrain control strategy (PT control) provides a target engine torque and transforms it into a engine load signal (0...1) depending on the current engine torque.
<b>Speed controller P/I -value (-)</b>	These parameters correspond to the proportional ( <i>P-value</i> ) and integral ( <i>I-value</i> ) of the rotation speed controller which mainly is a PI-controller. The speed controller becomes active if the powertrain control strategy (PT control) provides a target engine rotation speed and transforms it into a engine load signal (0...1) depending on the current engine rotation speed.
<b>Fuel cut-off (rpm)</b>	<p>This parameter sets the minimum engine rotation speed for which fuel cut-off becomes active during the coasting mode.</p> <p>More information about the ECU model can be found in <a href="#">Reference Manual chapter 'Engine control unit (ECU)'</a>.</p>

### 5.20.3 Parametrizing the TCU

One transmission control unit (TCU) is responsible for shifting all gearboxes and open/close all clutches of the powertrain.

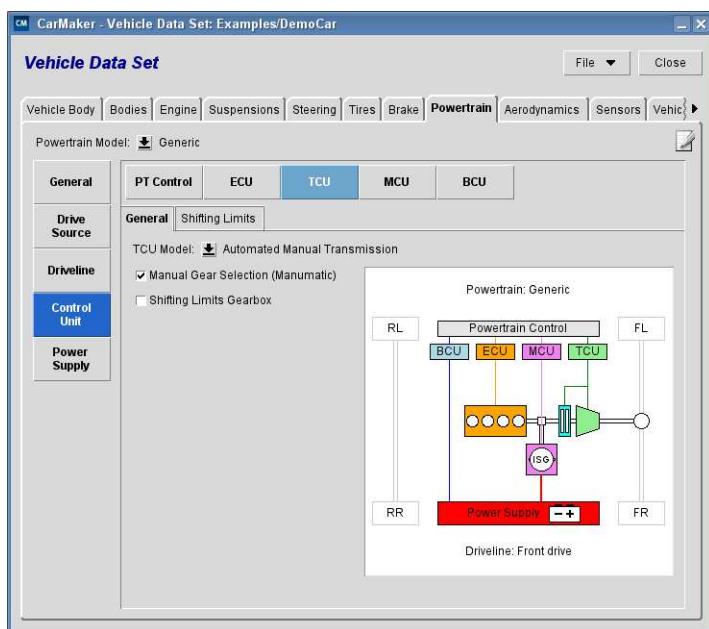


Figure 5.84: Selecting a TCU model

**TCU Model** CarMaker knows four different TCU models. Each one is specialized in one gearbox type (compare to [section 5.18.4 'Parametrizing the Gearbox'](#)).:

Table 5.79: TCU Models

Mode	Description
Automatic	TCU for the gearbox <i>Automatic</i> with an external hydraulic converter as clutch (only one gearbox)
Automatic with Converter	TCU for the gearbox <i>Automatic with Converter</i> with an internal hydraulic converter as clutch
Automated Manual Transmission	TCU for the gearbox <i>Automated Manual Transmission</i> with an internal friction clutch

Table 5.79: TCU Models

Mode	Description
not specified	No TCU model (e.g. for gearbox model <i>Manual</i> that is shifted by the driver or power split hybrids with a planetary gear)

**Manual Gear Selection (Manumatic)** You can activate manumatic for automatic gearboxes (*Automatic*, *Automatic with Converter* and *Automated Manual Transmission*) that belong to the same Drive Source as the combustion engine. Then it is possible to also shift the gears "manually" by IPGDriver or Direct Variable Access to the quantity *DM.GearNo*, as long as *DM.SelectorCtrl* is set to manual mode (2).

**Shifting Limits Gearbox** By deselecting this option, you can deactivate the TCU shifting control for each gearbox of your powertrain separately.

### TCU Model: Automatic

This TCU model for *Automatic* gearbox consists of two parts: *Lock-up Logic* and *Shifting Limits*.

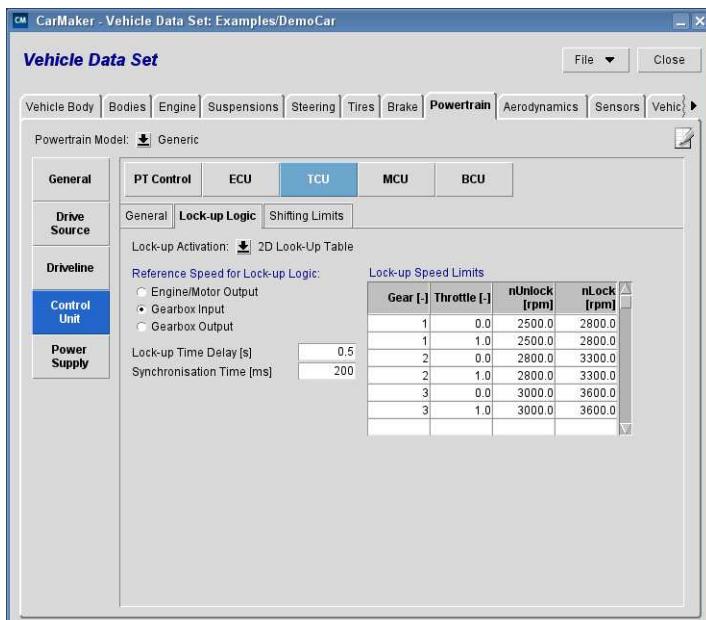


Figure 5.85: Lock-up Logic parametrization

The **Lock-Up Logic** defines when the lock-up clutch of the hydraulic converter is opened (converter unlocked) or closed (converter locked).

**Lock-up Activation** You can choose different activation modes for the lock-up logic:

Table 5.80: Lock-up Activation

Mode	Description
2D Look-up Table	This model gives you the possibility to define the rotations at which the converter is locked or unlocked for each gear in dependence on the throttle position.
DVA	No lock-up logic is parameterized: the converter is locked and unlocked during the simulation by the user through the DVA interface.

### Lock-up Activation: 2D Look-Up Table

#### Lock-up Speed Limits

This table gives you the possibility to define the rotation speeds at which the converter is locked ( $nLock$ ) or unlocked ( $nUnlock$ ) for each gear (Gear) in dependence on the throttle position (Throttle). The clutch is locked if the reference speed overruns the lock speed limit and is unlocked if the gear number changes or the reference speed underruns the unlock speed limit.

#### Reference Speed for Lock-Up Logic

You can select to which rotation speed the table *Lock-up Speed Limits* (column  $nUnlock$  and  $nLock$ ) refers to. The following components can be used:

- Engine Output shaft
- Gearbox Input shaft
- Gearbox Output shaft

#### Lock-up Time Delay (s)

Specifies the minimal time between the lock and unlock operation.

#### Synchronization Time (ms)

The synchronization time specifies the time to close the lock-up clutch and synchronize the rotation speeds of converter input and output side.

### Lock-up Activation: DVA

The DVA lock-up model is for very specific use. In this mode, the lock-up clutch can be controlled either directly by the user, or by an external model (see Programmer's Guide and [section 6.3 'DVA: Online Manipulation of the Simulation' on page 357](#) in this manual). To lock or unlock the converter, write to the User Accessible Quantity *PT.TCU.LockUp*.

More information about the DVA mechanism itself can be found in this document, [section 6.3 on page 357](#).

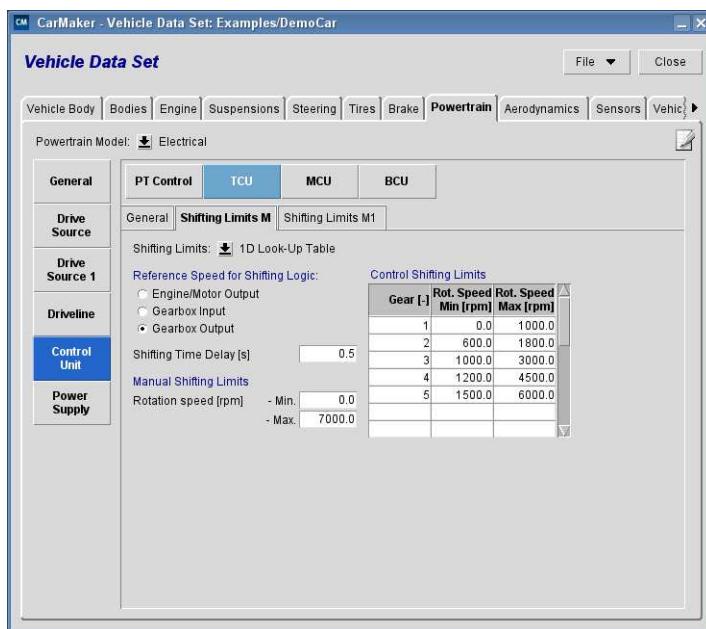


Figure 5.86: Shifting Limits parametrization

For each activated gearbox of the powertrain, the **Shifting Limits** are set by the following parameters.

**Shifting Limits** The rotation speeds for gear shifting can be set in two ways:

Table 5.81: Shifting Limits

Mode	Description
1D Look-Up Table	Shifting logic depends on the minimum and maximum allowed rotational speeds for each gear.
2D Look-Up Table	Shifting logic depends on the minimum and maximum allowed rotational speeds for each gear and additionally on the throttle position.

Both modes have similar parameters.

- Control Shifting Limits** This table defines the minimum (*Rot. Speed Min*) and maximum (*Rot. Speed Max*) rotation speed for each gear (*Gear*) in case of mode *1D Look-Up Table* or each combination of gear (*Gear*) and throttle position (*Throttle*). The values for throttle are between 0 and 1.
- Reference Speed for Shifting Logic** The reference rotation speed that you select here, applies to the minimum and maximum rotation speeds defined in the table *Control Shifting Limits*. You can choose between
- Engine Output shaft
  - Gearbox Input shaft
  - Gearbox Output shaft
- Shifting Time Delay (s)** This parameter states the minimal time between two shifting operations.
- Manual Shifting Limits** Here the engine rotation speed range during the manual shifting process (manumatic) is defined. If the engine speed reaches the limits and the driver does not react, the gearbox shifts itself up or down.

### TCU Model: Automatic with Converter

The parameters of this TCU are the same as for TCU model *Automatic* (see [section 'TCU Model: Automatic'](#)). The only difference is that here the lock-up logic applies to an internal clutch whereas the lock-up logic of *Automatic* applies to an external clutch.

### TCU Model: Automated Manual Transmission

This TCU can control several automated manual transmissions at a time. For each gearbox the shifting limits are defined the same way as for TCU *Automatic* (see [section 'Shifting Limits Gearbox'](#)). The TCU opens the clutch during the gear shifting process.

A description of all TCU models as well as the interfaces for user defined TCU models can be found in [Reference Manual chapter 'Transmission control unit \(TCU\)'](#).

## 5.20.4 Parametrizing the MCU

The motor control unit (MCU) contains the controllers of all electric motors, especially the load controllers. Each load controller provides a load signal for one electric motor based on the target rotation speed or target torque provided by PT Control and the motor's current rotation speed or torque.

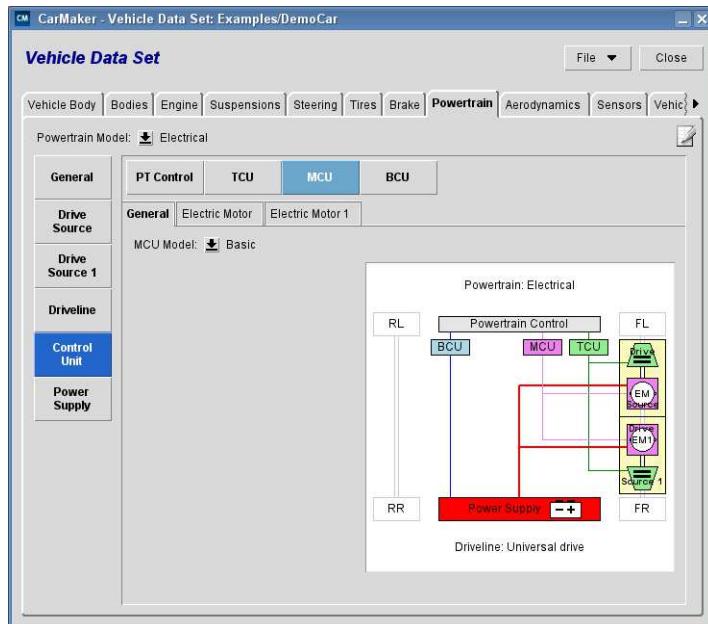


Figure 5.87: Selecting a MCU model

**MCU Model** Two MCU models are available in CarMaker:

Table 5.82: MCU Models

Mode	Description
Basic	Basic MCU model with PI controllers for load control
not specified	No MCU model (e.g. for generic powertrain)

### MCU Model: Basic

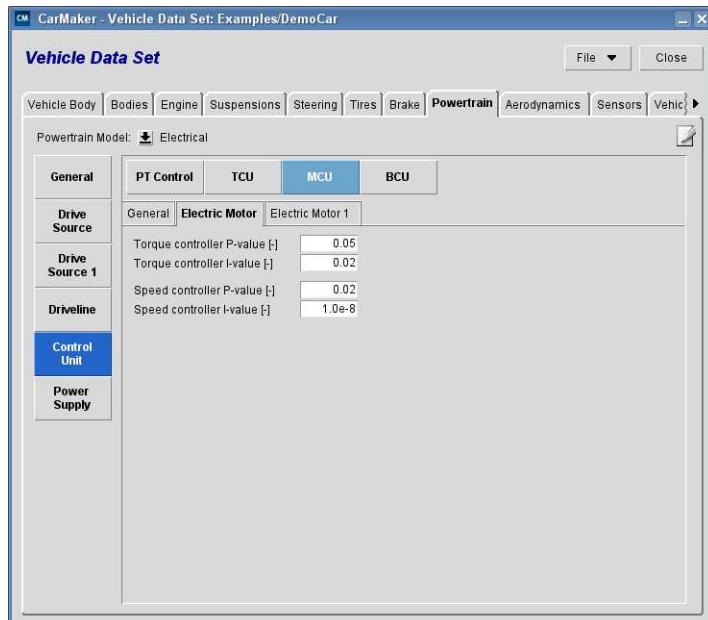


Figure 5.88: MCU model basic parametrization

**Torque controller P/I -value (-)** These parameters correspond to the proportional (*P-value*) and integral (*I-value*) of the torque speed controller which mainly is a PI-controller. The torque controller becomes active if the powertrain control strategy (PT control) provides a target mot torque and transforms it into a motor load signal (-1...1) depending on the current motor torque. A negative load signal means generator mode.

**Speed controller P/I -value (-)** These parameters correspond to the proportional (*P-value*) and integral (*I-value*) of the rotation speed controller which mainly is a PI-controller. The speed controller becomes active if the powertrain control strategy (PT control) provides a target motor rotation speed and transforms it into a motor load signal (-1...1) depending on the current motor rotation speed. A negative load signal means generator mode.

More information about the MCU model can be found in [Reference Manual chapter 'Motor control unit \(MCU\)'](#).

## 5.20.5 Parametrizing the BCU

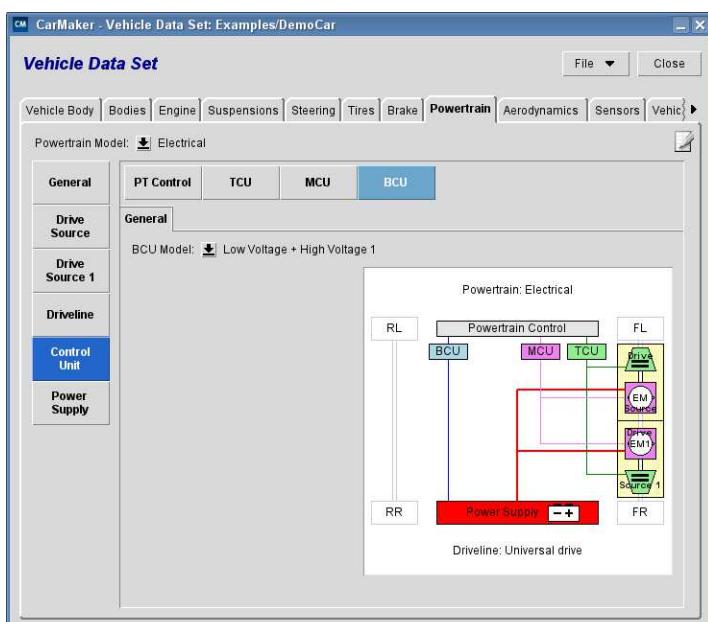


Figure 5.89: Selecting a BCU model

The main task of the battery control unit (BCU) is the calculation of characteristic battery values that cannot be measured directly such as state of charge (in %).

**BCU Model** CarMaker has four predefined BCU models each one specialized for a power supply configuration (compare [section 5.21 'Powertrain: Power Supply'](#)):

Table 5.83: BCU Models

Mode	Description
Low Voltage	BCU for a power supply consisting of a low voltage circuit with battery only
Low Voltage + High Voltage 1	BCU for a power supply consisting of a low voltage circuit and one high voltage circuit (two batteries)

Table 5.83: BCU Models

Mode	Description
Low Voltage + High Voltage 1 + High Voltage 2	BCU for a power supply consisting of a low voltage circuit and one high voltage circuit with battery and one high voltage circuit without battery
not specified	No BCU model (e.g. for generic powertrain without power supply)

For more information about BCU models, please refer to [Reference Manual chapter 'Battery control unit \(BCU\)'](#).

## 5.21 Powertrain: Power Supply

The power supply consists of the vehicles electric circuits, their batteries and auxiliaries consumers.



CarMaker's power supply module follows a power based approach. That means, for each circuit motor, generator and auxiliary electric power is summed up to get the battery load. The electric current and voltage is not calculated separately for each wire inside the circuits. [Reference Manual chapter 'Power Supply'](#) provides additional information about this approach.

### 5.21.1 Parametrizing the Power Supply

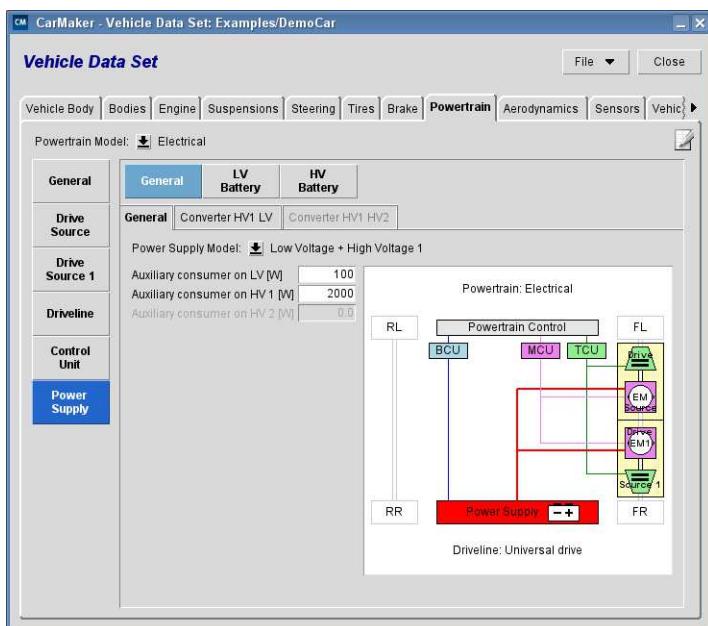


Figure 5.90: Selecting a power supply model

#### Power Supply Model

CarMaker contains four different power supply configurations:

Table 5.84: Power Supply Models

Model	Description
Low Voltage	Power supply consisting of a low voltage circuit with battery only
Low Voltage + High Voltage 1	Power supply consisting of a low voltage circuit with battery and one high voltage circuit with battery
Low Voltage + High Voltage 1 + High Voltage 2	Power supply consisting of a low voltage circuit with battery, one high voltage circuit with battery and one high voltage circuit without battery
not specified	No power supply model (e.g. for generic powertrain with simple starter model)

#### Auxiliary consumers on LV/HV1/HV2 (W)

You can define a constant load on each electric circuit to take into account the electric power consumed (or generated) by auxiliary devices such as air conditioning. Positive consumption values decrease the available electric energy inside the circuit, negative values increase the electric energy available.



It is possible to modify the auxiliary consumer power on each circuit during the TestRun. this can be done inside a user defined model (e.g. of a air conditioning) or via Direct Variable Access. In both cases, the electric power should be written on the quantities *PT.PwrSupply.LV.Pwr\_aux*, *PT.PwrSupply.HV1.Pwr\_aux* and *PT.PwrSupply.HV2.Pwr\_aux*. The sign convention is the same as for constant auxiliary consumption.

## Power Supply Model: Low Voltage

This model has no additional parameters except the parameters of the low voltage battery described in [section 5.21.2 'Parametrizing the batteries'](#).

## Power Supply Model: Low Voltage + High Voltage 1

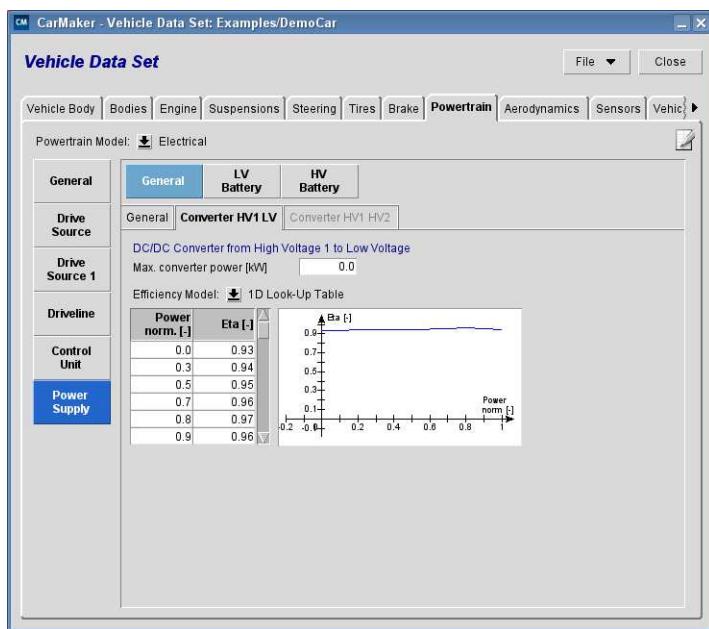


Figure 5.91: Power Supply LV+HV1 parametrize the DC/DC converter

The low and high voltage circuit of this power supply are interconnected via a DC/DC converter whose characteristics are defined in the tab *Converter HV1 to LV*.

### Max. converter power (kW)

Here you can set the maximum electric power that can be transferred between low and high voltage circuit. A maximum converter power of 0kW means, that the circuits are not connected.

### Efficiency Model

There are two possibilities to describe the efficiency of the energy transfer:

Table 5.85: DC/DC converter Efficiency models

Mode	Description
Characteristic Value	Constant efficiency
1D Look-Up Table	Efficiency depends on the exchanged power

### Efficiency Model: Characteristic Value

**Efficiency (-)** The efficiency of the DC/DC converter is constant.

### Efficiency Model: 1D Look-Up Table

The efficiency of the DC/DC converter ( $\text{Eta}$ ) depends on the transmitted power divided by the maximum converter power ( $\text{Power norm.}$ ).

For the parametrization of the low and high voltage batteries inside this power supply, please refer to [section 5.21.2 'Parametrizing the batteries'](#).

## Power Supply Model: Low Voltage + High Voltage 1 + High Voltage 2

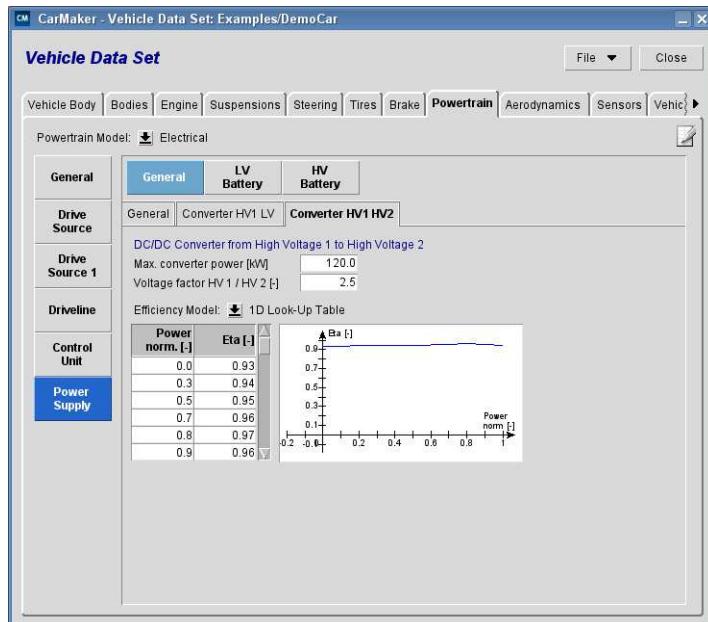


Figure 5.92: Power Supply LV+HV1+HV2 parametrize the DC/DC converter

This power supply corresponds to *Low Voltage + High Voltage 1* with an additional high voltage circuit that is connected via another DC/DC converter. It does not contain an extra battery. The *Converter HV1 LV* and the low and high voltage battery need the same parameters as for *Low Voltage + High Voltage 1*. The *Converter HV1 HV2* is based on *Converter HV1 LV* with one additional parameter:

- Voltage factor HV1/HV2 (-)** This factor defines the voltage level of high voltage circuit 2 relative to high voltage circuit 1.

### 5.21.2 Parametrizing the batteries

The power supply may contain one or two batteries. CarMaker supports the same models for low and high voltage batteries.

- Battery Model** Currently one battery model is available in CarMaker:

Table 5.86: Battery models

Mode	Description
Chen	The battery model of Chen consists of an electric battery model combined with a characteristic curve to take into account the influence of state of charge on the voltage

## Battery Model: Chen

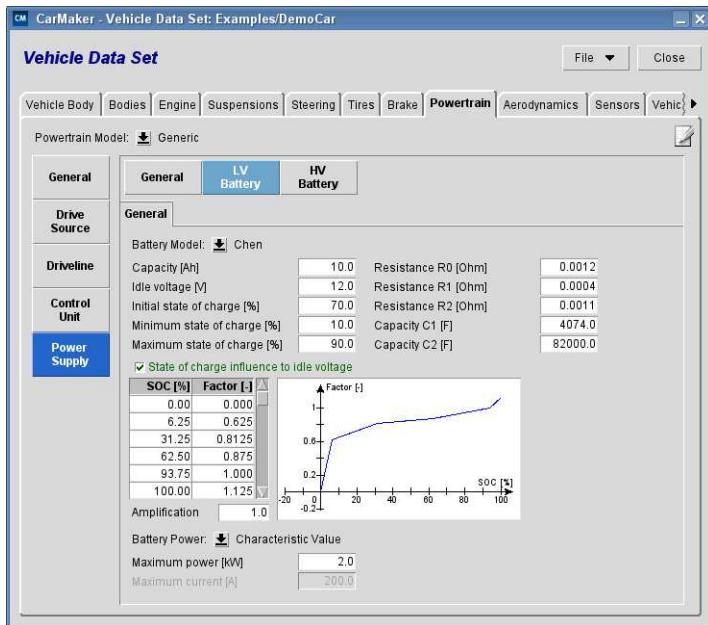


Figure 5.93: Battery Model Chen

The battery's equivalent circuit consists of a ideal voltage source, two RC-circuits and a single resistance that are connected in series as shown in [Figure 5.94](#).

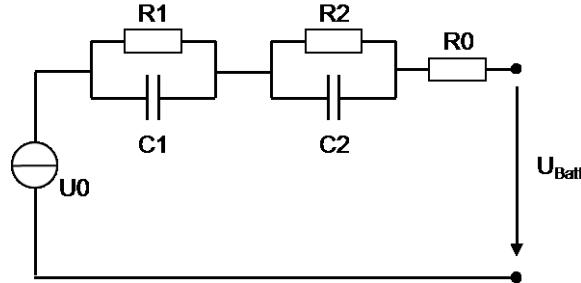


Figure 5.94: Chen battery equivalent circuit

- |                                    |  |
|------------------------------------|--|
| <b>Capacity (Ah)</b>               | This sets the overall capacity of the battery.   |
| <b>Idle voltage (V)</b>            | This parameter is the basic voltage of the ideal voltage source ( $U_{oc}$ ) as shown in <a href="#">Figure 5.94</a> without any influence of SOC. |
| <b>Initial state of charge (%)</b> | The initial state of charge defines the batteries load at simulation start in %.   |
| <b>Minimum state of charge (%)</b> | The batteries state of charge should not fall below this limit because this would damage the battery.  |
| <b>Maximum state of charge (%)</b> | The batteries state of charge should not exceed this limit because this would damage the battery.  |
| <b>Resistance R0/R1/R2 (Ohm)</b>   | Here you can set the resistances $R_0$ , $R_1$ and $R_2$ of the equivalent circuit shown in <a href="#">Figure 5.94</a> .                          |

<b>Capacity C1/C2 (F)</b>	Here you can set the capacities $C_1$ and $C_2$ of the equivalent circuit shown in <a href="#">Figure 5.94</a> .
<b>State of charge influence to idle voltage</b>	If this option is selected, you can set the influence of the battery's state of charge (SOC) on its idle voltage. It is defined as a factor ( <i>Factor</i> ) on $U_0$ .
<b>Amplification (-)</b>	This amplification factor applies to all values in the column <i>Factor</i> , if <i>State of charge influence to idle voltage</i> is selected. It enables to modify the SOC influence quickly without editing every single value of the table.
<b>Battery Power</b>	There are two modes to limit three modes to limit the battery charging and discharging power at a time:

Table 5.87: Battery power modes

Mode	Description
Constant	The battery's charging/discharging power is limited by a maximum power.
By Current	The battery's charging/discharging power is limited by a maximum electric current.
DVA	The battery's charging/discharging power limit is not calculated inside the model but provided via DVA.

**Battery Power: Constant**

**Maximum power (kW)** The battery cannot be charged or discharged by more than this maximum power.

**Battery Power: By Current**

**Maximum current (A)** The battery cannot be charged or discharged by more than this electric current.

**Battery Power: DVA**

The DVA power mode is for very specific use. In this mode, the maximum charge and discharge power is set either directly by the user, or by an external model (see Programmer's Guide and [section 6.3 'DVA: Online Manipulation of the Simulation' on page 357](#) in this manual). The User Accessible Quantity `PT.BattLV.Pwr_max` and `PT.BattHV.Pwr_max` need to be used.

For further details about the battery model, please refer to [Reference Manual chapter 'Battery'](#).

## 5.22 Aerodynamics

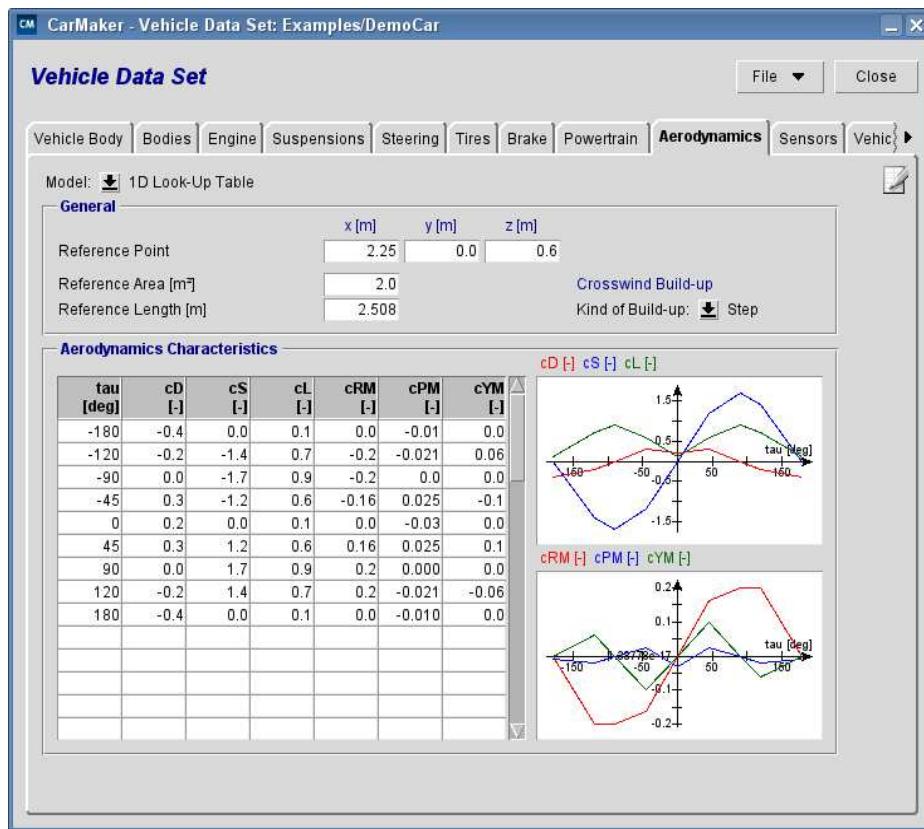


Figure 5.95: Definition of the aerodynamics parameters

The aerodynamic model applies additional variable forces to the vehicle body according to the vehicle speed. The adaptation of the map by accurate values is often not required as the aerodynamic model is of minor importance at low speeds. Under about 100 km/h the influence of the aerodynamic forces is so small that you can afford to leave the default values.

If you need a better aerodynamic model that takes into account the ride height, please contact your sales partner at IPG to let you know about a more detailed aerodynamic model. However, this model already enables to reach very good results, and that is why it has been used successfully even by our customers working on racing applications.

**Model** By default, only one aerodynamic model is available in CarMaker. The single default option available is the model *Look-Up Table 1D*. Additional information about the aerodynamical model, especially about the equations used to calculate the aerodynamic forces, can be found in the [Reference Manual, section 'Aerodynamics'](#).

In case you would like to use your own aerodynamic model (built in Simulink or C-Code), or you would like to exclude the aerodynamic influences on your vehicle completely, the option *not specified* should be chosen for *Model*.

The following general parameters need to be defined for the aerodynamic model *Look-Up Table 1D* (all coordinates refer to FrameD (see [section 'Origin Fr1: the Coordinate System \[m\]](#) on page 154 for information about the axis systems)):

**Aero Marker (m)** Caution: it is the single aerodynamical parameter that is to be parameterized in the tab *Bodies!* This point is used by the module that simulates side winds: see [section 'Wind' on page 62](#).

When the vehicle is gripped by side wind, the wind starts to take effect only from a certain point on, e.g. if only the bumper is attacked by side wind the driver will usually not recognize the effects. Ahead of this point, the vehicle body does not offer enough contact surface to the wind to take effect.

It is that very point that is parameterized by the *Aero Marker*. Please note that it is not the point where the aerodynamic forces apply! It is only the point from which the forces created by the side winds are taken into account. But the forces themselves are applied to another point (see Reference Point [m]).

We suggest to select this point according to the shape of the vehicle body, e.g at the level of the front wheel for a van, and just in front of the cockpit for a vehicle with a long hood such as a typical sports car.

**Reference Point (m)** It is the center of pressure, the point where the aerodynamic forces, including the side wind forces, are applied.

If you do not have any information on the exact location of this point, the vehicle's center of gravity can be taken instead. According to the shape of the vehicle body, you can move this point so that the aerodynamic influence in the simulation approaches to what you observe on the track.

**Reference Area (m<sup>2</sup>)** The vehicle *Reference Area* is the projected frontal area including tires and underbody parts.

**Reference Length (m)** Per default the *Reference Length* is the same as the wheel base of the vehicle. It is used to calculate the torques applied to the Reference Point due to aerodynamic effects. If you wish to tune this value, have a look at the equations in the [Reference Manual](#).

**Aerodynamics Coefficients (-)** The *Aerodynamic Coefficients* used by the model *Look-Up Table 1D* have the following meaning:

Table 5.88: Aerodynamic Coefficients

Parameter	Description	Unit
tau	Wind angle	deg
cD	Coefficient used to calculate the aerodynamical drag force (x-axis).	-
cS	Coefficient used to calculate the aerodynamical side force (y-axis).	-
cL	Coefficient used to calculate the aerodynamical lift force (z-axis).	-
cRM	Coefficient used to calculate the aerodynamical roll moment (around x-axis).	-
cPM	Coefficient used to calculate the aerodynamical pitch moment (around y-axis).	-
cYM	Coefficient used to calculate the aerodynamical yaw moment (around z-axis).	-

Please find more information about the formulas that use above coefficients in the [Reference Manual](#), section '*Aerodynamics*'.

The values used for a zero wind angle are probably the most important ones since they will always be used during the simulation: driving the vehicle causes a virtual front wind. Other angle values are used especially if side wind occurs ([section 'Wind' on page 62](#)). In particular, the value of the parameter  $cD$  for  $\tau = 0$  deg is the well known  $c_x$  value.

## 5.23 Sensors

In the *Sensors* tab, there is the possibility to define different types of sensors. They can be classified into three main sensor groups depending on their functionality. *Side Slip Angle Sensors* and *Body Sensors* measure the vehicle behavior. *Driver Assistance Sensors* and *Free Space Sensors* detect traffic objects. *Traffic Sign Sensors*, *Line Sensors* and *Road Property Sensors* detect features of the road (road characteristics or sensor objects). The Collision Detection is not a classical sensor, but monitors collisions between the ego vehicle and other traffic objects.

### 5.23.1 Side Slip Angle Sensors

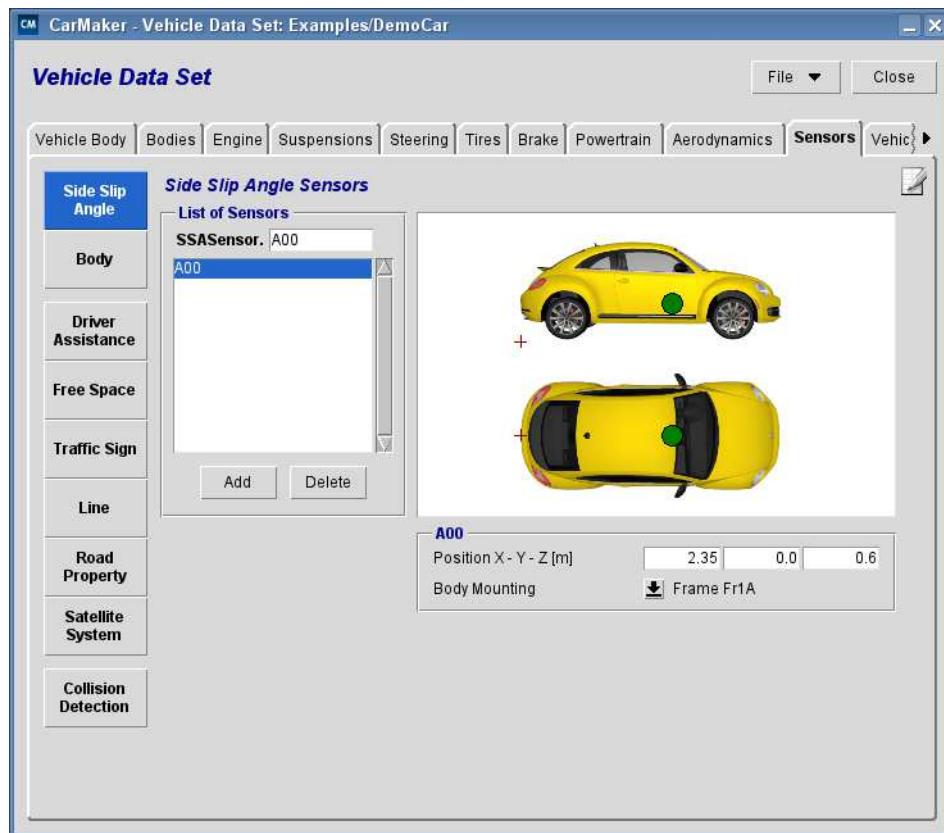


Figure 5.96: Definition of *Side Slip Angle Sensors*

You have the possibility to define up to 30 *Side Slip Angle Sensors* (total number of this sensor kind in vehicle and trailer). At least one must be specified, which is located in the vehicle's center of gravity per default.

The parameters of the *Side Slip Angle Sensor* are as follows:

- SSASensor.** Here you can enter a name for the current sensor, to alleviate the identification of the sensor quantities, e.g. in IPGControl. All User Accessible Quantities of this sensor will start with the prefix *Car.SSASensor.Name.\**. In the example in [Figure 5.96](#) the UAQs of this sensor will start with *Car.SSASensor.A00.\**.
- Position X-Y-Z (m)** The location of the *Side Slip Angle Sensor* is defined by coordinates in *FrameD* (see [section 'Origin Fr1: the Coordinate System \[m\]](#) on page 154 for information about the axis systems). The sensor position is visualized on the graphical representation of the vehicle.
- Body Mounting** The mounting point of the *Side Slip Angle Sensor* is defined here. This coordinates are always provided in the vehicle based coordinate system *Fr1A*. Nevertheless you can choose if the sensor is connected to *Fr1A* or *Fr1B* (see [section 5.2 'CarMaker Coordinate Systems'](#) on page 148 for information about the axis systems) in case a *Flexible Body* is used ([section 5.3 'Bodies'](#) on page 150).

## 5.23.2 Body Sensors

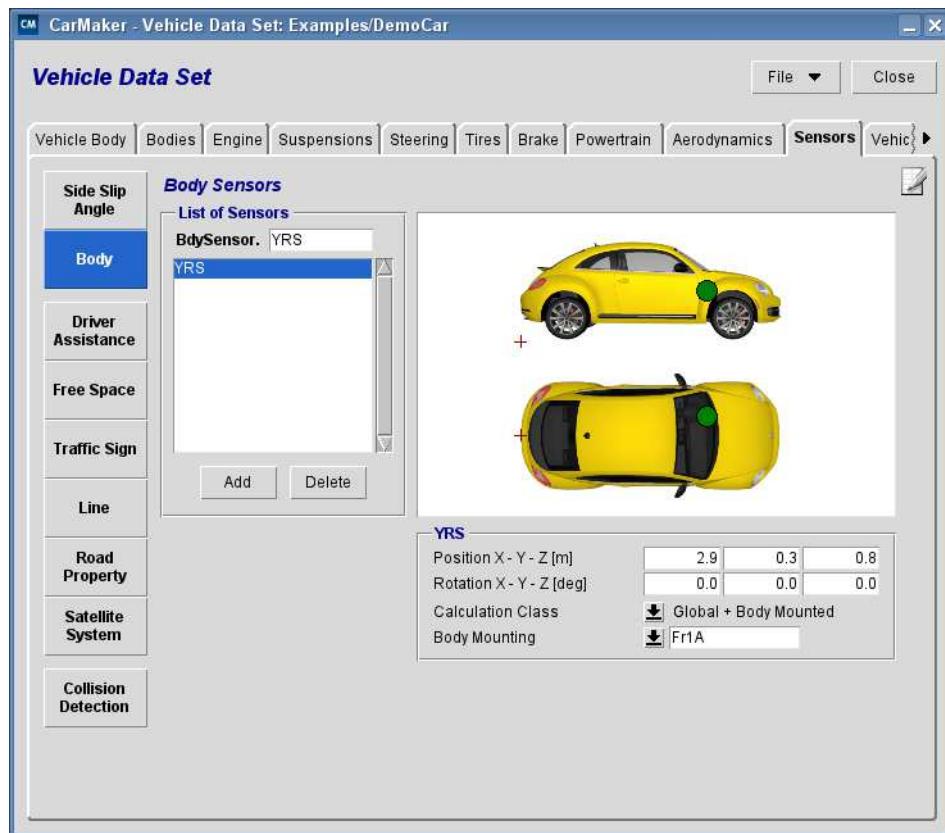


Figure 5.97: Definition of *Body Sensors*

A *Body Sensor* is an inertial sensor that can be positioned anywhere on the vehicle to measure the position, velocity, rotational velocity, acceleration and rotational acceleration. Up to 30 *Body Sensors* can be placed in total on the vehicle and the trailer. A small set of parameters enables to parameterize each of them:

- BdySensor.** Here you can enter a name for the current sensor, to alleviate the identification of the sensor quantities, e.g. in IPGControl. All User Accessible Quantities of this sensor will start with the prefix *BodySensor.Name.\**. In the example in [Figure 5.97](#), the UAQs of this sensor will start with *BodySensor.YRS.\**.

**Position X-Y-Z (m)** These coordinates indicate the position of the selected sensor on the vehicle. They are interpreted in the *FrameD* (see section '[Origin Fr1: the Coordinate System \[m\]](#)' on page [154](#) for information about the axis systems). If the body sensor is mounted on the wheel carrier, the parameters describe the position before the static equilibrium configuration. The sensor position is visualized on the graphical representation of the vehicle.

**Rotation X-Y-Z (deg)** These angles specify the rotation of the sensor frame according to the frame on which the sensor is mounted. Rotation order: ZYX.

**Calculation Class** This parameter specifies which frame should be used as reference for the calculation of displacements, velocity and acceleration in the sensor position. The following options are available:

Table 5.89: Description of the *Calculation Classes*

Calculation Class	Description
Global	All velocities and accelerations are calculated in global frame Fr0
Global + Body Mounted	All velocities and accelerations are calculated in the global frame Fr0 and in the local frame on which the body sensor is mounted.
Global + Body Mounted no G	All velocities and accelerations are calculated in the global frame Fr0 and in the local frame on which the body sensor is mounted. The accelerations in the local frame do not consider the gravitation of the earth.

**Mounting** The parameter indicates the frame on which the selected body sensor is mounted. A detailed description on the meaning of the frames can be found in [section 5.2 on page 148](#).

Table 5.90: Description of the *Mounting* positions

Mounting	Description
Frame Fr1A/B	Mounted on the main body; in case of a flexible body the sensor can be mounted either on the front or rear body part (see section <a href="#">5.3 on page 150</a> ).
Frame FrEng	Mounted in the engine body; this option is available only if an elastic mounted engine is used.
Frame Fr2<pos>	Mounted on the wheel carrier frames, where <pos> stands for the wheel position (FL=front left, FR=front right, RL=rear right, RR=rear right).

For further details about Body Sensors, see the [Reference Manual](#), section '[Body Sensors](#)'.

### 5.23.3 Driver Assistance Sensors

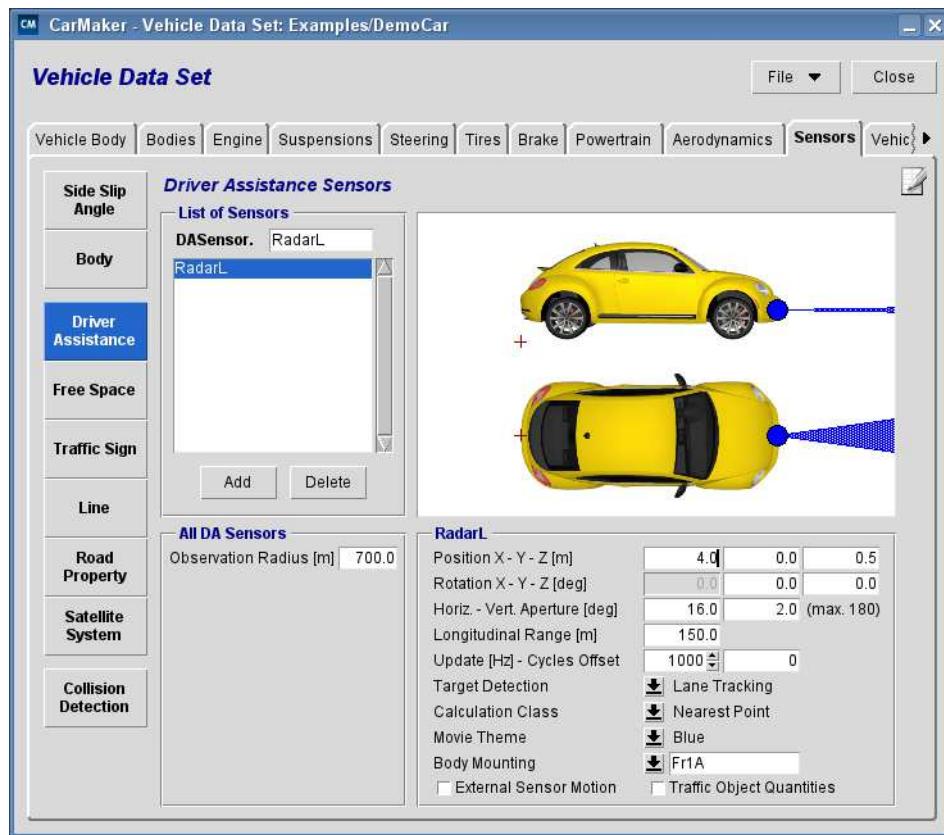


Figure 5.98: Definition of *Driver Assistance Sensors*

CarMaker offers the possibility to integrate up to 30 *DriverAssistance Sensors* in the vehicle. Each one can be parameterized independently, just by selecting the sensor in the *General* area. The parameters that are specific to the selected sensor are displayed under the graphical representation of the vehicle.

**DASensor.** Here you can enter a name for the current sensor, to alleviate the identification of the sensor quantities, e.g. in IPGControl. All User Accessible Quantities of this sensor will start with the prefix *DASensor.Name.\**. In the example in Figure 5.98, the UAs of this sensor will start with *DASensor.DA00.\**.

**Observation Radius (m)** To avoid the calculation of far-off objects, an observation area around the sensor with a constant radius is defined. All quantities are calculated and updated only if the object is located in the observation area. Default: 700m.

**Position X-Y-Z (m)** These coordinates indicate the position of the selected sensor on the vehicle. They are interpreted in the *FrameD* (see section '[Origin Fr1: the Coordinate System \[m\]](#)' on page 154 for information about the axis systems). The sensor position is visualized on the graphical representation of the vehicle.

**Rotation X-Y-Z (deg)** These angles specify the rotation of the sensor frame according to the vehicle frame (*FrameD*). This specifies the viewing direction of the sensor. Rotation order: ZYX.

**Horizontal Aperture (deg)** This parameter defines the horizontal aperture of the sensor beam [deg], the maximum angle is 180 °deg.

<b>Vertical aperture (deg)</b>	This parameter defines the vertical aperture of the sensor beam [deg], the maximum angle is 180 °deg.
<b>Longitudinal Range (m)</b>	The longitudinal range of the sensor beam is defined here.
<b>Update (Hz)</b>	This parameter defines the frequency, with which the environment is scanned by the driver assistance sensor.
<b>Cycles Offset</b>	This parameter serves for performance optimization of the simulation. If you are using more than one sensor and not every ms is required for simulation, you can define an offset here.
<b>Target Detection</b>	This parameter specifies the sensor type:

Table 5.91: *Target Detection* modes for the *Driver Assistance Sensors*

Mode	Description
Lane Tracking	The detection of a relevant target is done by investigating the current vehicle trajectory, the width of its driving lane and the distance to the object.
Nearest Object	The sensor module finds the closest object and calculates the distance from this object to the sensor. The object with the smallest overall distance is the relevant target.
None	No target detection.



Please note, that hill tops or fog will have no effect on the range of the sensor.



As an example take the TestRun Examples > CarMakerFunctions > Sensors > DASensor\_LaneTracking.

<b>Movie Theme</b>	At this point, you can define the color of the sensor beam which is used for visualization purposes in IPGMovie. It does not have any effect on the detection of obstacles.
<b>Calculation Class</b>	This parameter specifies if the calculated distance to the object refers to the reference point or to the nearest point:

Table 5.92: Different *Calculation Classes* for the *Driver Assistance Sensors*

Mode	Description
Reference Point	Calculation of relative quantities in reference point.
Relative Angles	Calculation of relative quantities in reference point and calculation of relative angle between the object and the ego car (comparison of x-axis Fr1).
Nearest Point	Calculation of relative quantities in reference point (including relative angles) AND in nearest point.
Image Area	Calculation of relative quantities in reference point (including relative angles) AND in nearest point; calculation of incidence angles and projections on image area.

**Additional Traffic Object Quantities** If checked, additional quantities for all traffic objects and not only the detected obstacles are available in the *Data Dictionary*. For example, you can access those quantities via IPGControl. Be aware that this may increase the number of User Accessible Quantities a lot.

**Consider External Sensor Motion** If checked, external relative sensor travel and rotation are considered. To use the external sensor motion, the DVA-quantities *DASensor.<Sensor-Name>.rx\_ext*, *DASensor.<Sensor-Name>.ry\_ext* and *DASensor.<Sensor-Name>.rz\_ext* have to be defined.

For more information about the external sensor motion, please have a look at section '[Parametrization of DASensors](#)' in the Reference Manual.

For further details about *Driver Assistance Sensors*, see the Reference Manual, section '*Driver Assistance Sensor*'.

## 5.23.4 Free Space Sensor

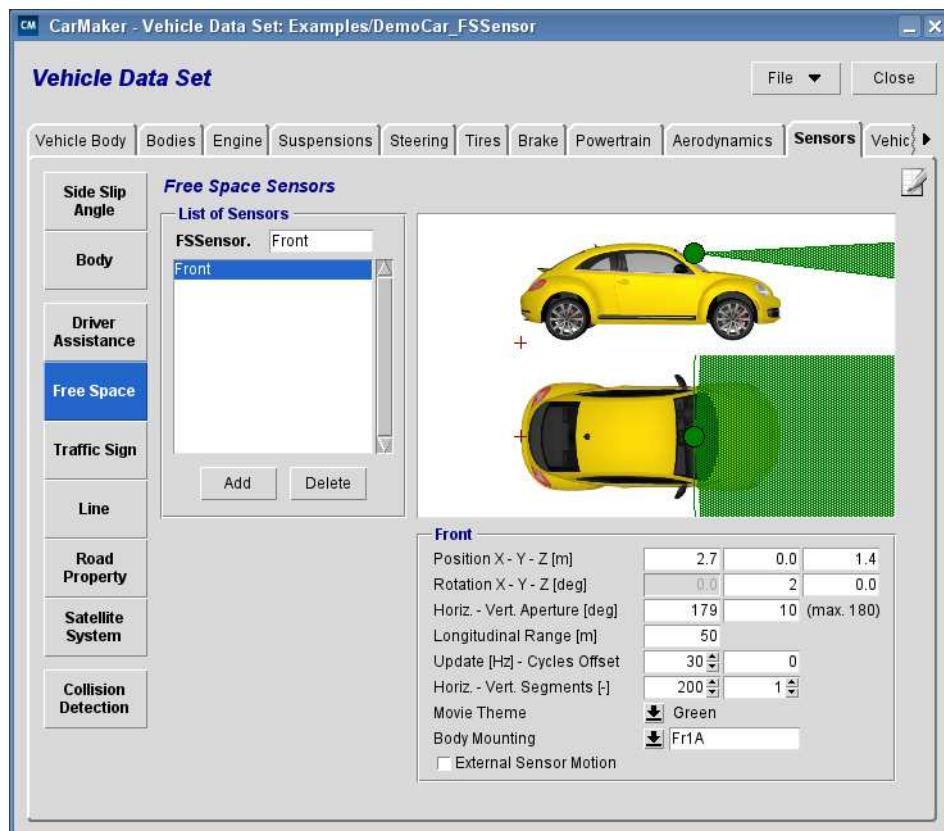


Figure 5.99: Definition of a *Free Space Sensor*

The *Free Space Sensor* module (FSSensor) is an extended DASensor module, whose sensor beam is subdivided in equiangular horizontal and vertical segments. Each segment determines the nearest detected point of the surrounding traffic objects and the corresponding bearing angles and approach velocity.

This sensor module is able to scan the environment like a radar sensor and capture the free and the occupied space around the vehicle, which could be further used for an avoidance maneuver assistant. In CarMaker, up to 30 sensors of this kind can be defined for each vehicle.

<b>FSSensor.</b>	Here you can enter a name for the current sensor, to alleviate the identification of the sensor quantities, e.g. in IPGControl. All User Accessible Quantities of this sensor will start with the prefix <code>FSSensor.Name.*</code> . In the example in <a href="#">Figure 5.99</a> , the UAQs of this sensor will start with <code>FSSensor.FS00.*</code> .
<b>Position X - Y - Z (m)</b>	These coordinates indicate the position of the <i>Free Space Sensor</i> on the vehicle. They are interpreted in the <i>FrameD</i> (see <a href="#">section 'Origin Fr1: the Coordinate System [m]' on page 154</a> for information about the axis systems). The sensor position is visualized on the graphical representation of the vehicle.
<b>Rotation X - Y - Z (deg)</b>	These angles specify the rotation of the sensor frame according to the frame on which the sensor is mounted. Rotation order: ZYX.
<b>Horiz. - Ver. Aperture (deg)</b>	This parameter defines the horizontal and vertical aperture of the sensor beam [deg], maximum angle is 180 deg.
<b>Longitudinal Range (m)</b>	The longitudinal range of the sensor beam is defined at this point.
<b>Horiz. - Ver. Segments (-)</b>	Defines the number of the horizontal and vertical sensor beam segments.
<b>Update (Hz) - Cycles Offset</b>	This parameter defines the frequency, with which the environment is scanned by the free space sensor.
<b>Movie Theme</b>	At this point, you can define the color of the sensor beam which is used for visualization in IPGMovie. It does not have any effect on the detection of obstacles.
<b>Body Mounting</b>	The mounting point of the <i>Free Space Sensor</i> is defined here. It is always the vehicle based coordinate system <i>Fr1A</i> , in case a <i>Flexible Body</i> is used you can choose between <i>Fr1A</i> and <i>Fr1B</i> (see <a href="#">section 5.2 'CarMaker Coordinate Systems' on page 148</a> for information about the axis systems and section <a href="#">section 5.3 'Bodies' on page 150</a> for more information on flexible bodies).
<b>External Sensor Motion</b>	When this tick box is activated, the external relative sensor travel and rotation is activated, and they can be manipulated by the corresponding UAQs <code>FSSensor.&lt;Sensor-Name&gt;.rx_ext</code> , <code>FSSensor.&lt;Sensor-Name&gt;.ry_ext</code> , <code>FSSensor.&lt;Sensor-Name&gt;.rz_ext</code> , <code>FSSensor.&lt;Sensor-Name&gt;.tx_ext</code> , <code>FSSensor.&lt;Sensor-Name&gt;.ty_ext</code> and <code>FSSensor.&lt;Sensor-Name&gt;.tz_ext</code> .

## 5.23.5 Traffic Sign Sensor

The *Traffic Sign Sensor* (TSSensor) detects all or pre-selected traffic signs which fall within its defined range and horizontal/vertical aperture angles. The sensor cross-checks if the sign faces the sensor and then sorts all detected signs in ascending order with distance. The sensor also provides additional information about the traffic sign such as identification, supplement sign information, sign description, and sign values (e.g. velocity).

Please notice, that this sensor only detects traffic signs that have been defined on the Road dialog using the Sensor Objects > Traffic sign option as a new Bump/Marker, but not the Marker > SpeedLimit and Stop. It is possible to define up to 5 *Traffic Sign Sensors* for each vehicle.

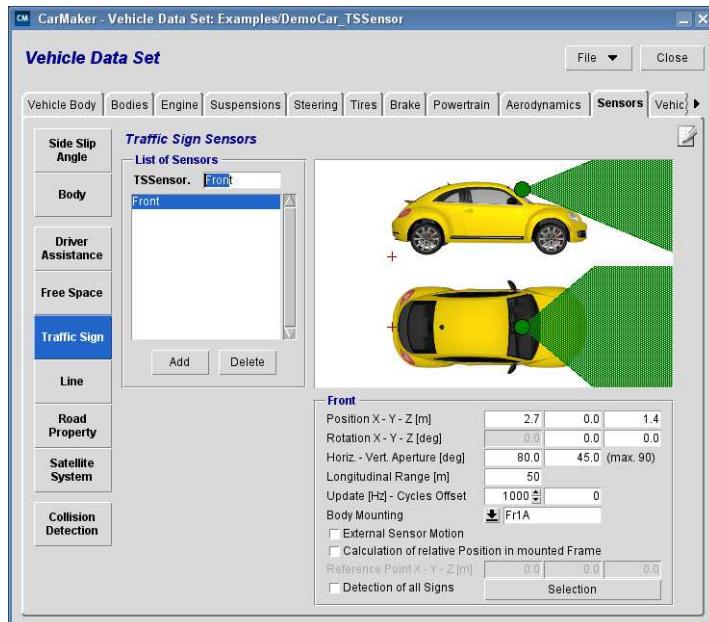


Figure 5.100: Traffic Sign Sensor parameterization

- TSSensor.** Here you can enter a name for the current sensor, to alleviate the identification of the sensor quantities, e.g. in IPGControl. All User Accessible Quantities of this sensor will start with the prefix *TSSensor.Name.\**. In the example in [Figure 5.100](#), the UAQs of this sensor will start with *TSSensor.TS00.\**.
- Position X - Y - Z (m)** These coordinates indicate the position of the Traffic Sign Sensor module on the vehicle. They are interpreted in *FrD* (see [section 'Origin Fr1: the Coordinate System \[m\]' on page 154](#) for information about the axis systems). The sensor position is visualized on the graphical representation of the vehicle.
- Rotation X - Y - Z (deg)** These angles specify the rotation of the sensor frame according to the frame on which the sensor is mounted. Rotation order: ZYX.
- Horiz. - Ver. Aperture (deg)** This parameter defines the horizontal and vertical aperture of the sensor beam [deg], maximum angle is 180 deg.
- Longitudinal Range (m)** The longitudinal range of the sensor beam is defined at this point.
- Update (Hz) - Cycles Offset** The parameter *Update [Hz]* defines the frequency with which the environment is scanned by the traffic sign sensor. The *Cycles Offset* is an option to provide an offset with respect to the number of simulation cycles from the start of the simulation after which the traffic sign sensor module should be activated.
- Body Mounting** The mounting point of the *Traffic Sign Sensor Module* is defined here. By default it is the vehicle based coordinate system *Fr1A*. In case a *Flexible Body* is used you can choose between *Fr1A* and *Fr1B* (see [section 5.2 'CarMaker Coordinate Systems' on page 148](#) for information about the axis systems and [section 5.3 'Bodies' on page 150](#) for more information on flexible bodies).

<b>External Sensor Motion</b>	When this tick box is activated, the external relative sensor travel and rotation is activated, and they can be manipulated by use of the corresponding UAQs <i>TSSensor.&lt;Sensor-Name&gt;.rx_ext</i> , <i>TSSensor.&lt;Sensor-Name&gt;.ry_ext</i> , <i>TSSensor.&lt;Sensor-Name&gt;.rz_ext</i> , <i>TSSensor.&lt;Sensor-Name&gt;.tx_ext</i> , <i>TSSensor.&lt;Sensor-Name&gt;.ty_ext</i> and <i>TSSensor.&lt;Sensor-Name&gt;.tz_ext</i> .
<b>Calculation of relative Position in mounted Frame</b>	Specifies if the relative sign positions are expressed and sorted relating to the body frame on which the sensor is mounted (e.g.: vehicle frame Fr1A) or the sensor frame.
<b>Reference Point - X- Y- Z (m)</b>	Position of the reference point in vehicle frame FrD in which the relative sign position is calculated (only if <i>Calculation of relative Position in mounted Frame</i> is chosen).  Default: 0 0 0.
<b>Detection of all Signs</b>	All the signs that fall with the sensor beam are detected if this option is selected. When this option is not selected, then all the required types of signs can be chosen by clicking on <i>Selection</i> .
<b>Selection</b>	The types of traffic signs that need to be detected can be chosen in this dialog box. Please note that the <i>Selection</i> button is grayed out until <i>Detection of all Signs</i> is deselected. The number of traffic sign selections is limited to 10 types.

## 5.23.6 Line Sensor

The *Line Sensor* module is used to detect lines and barriers that have been defined by the user on the Road dialog using the Road Marking and/or Traffic Barrier options as a new Bump/Marker. It can be compared to an idealized camera.

The sensor provides information such as identification code, color, type, width and height (of the Traffic Barrier). The sensor sorts the detected lines and barriers into left and right elements with respect to the origin of the sensor frame or any reference point that can be defined in the vehicle frame *Fr1* and then sorts them in ascending order with the lateral distance from the chosen frame.

A detected line is divided into a number of points which can be used e.g. to calculate the lane/path prediction. Please note, that the information about the relative position of these points are not provided as User Accessible Quantities by default. In fact, there are many different possibilities to define the distance of the vehicle to a line or a spline through the line points. The global header file *LineSensor.h* can be used to read the coordinates of the line points and implement your own algorithm using the CarMaker C-code interface.

It is possible to define up to 5 *Line Sensors* for each vehicle.

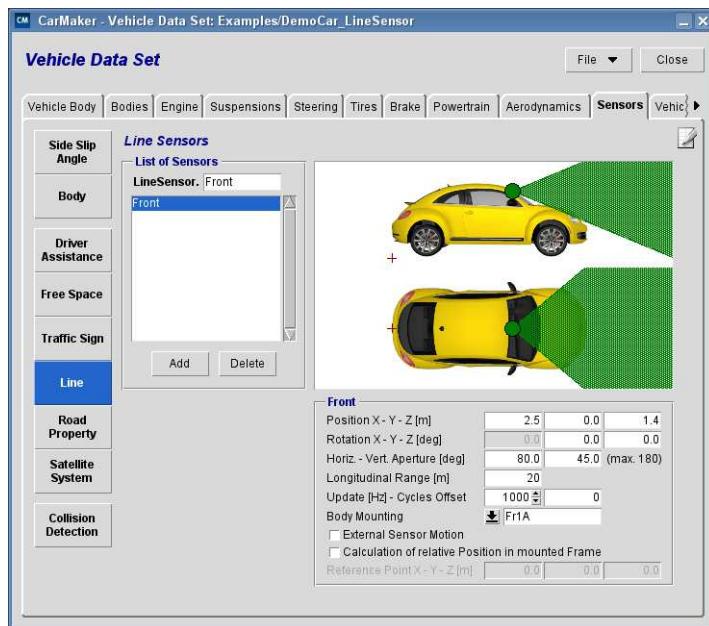


Figure 5.101: Line Sensor parameterization

**LineSensor.** Here you can enter a name for the current sensor, to alleviate the identification of the sensor quantities, e.g. in IPGControl. All User Accessible Quantities of this sensor will start with the prefix *LineSensor.Name.\**. In the example in [Figure 5.101](#), the UAQs of this sensor will start with *LineSensor.LN00.\**.

**Position X - Y - Z (m)** These coordinates indicate the position of the Line Sensor Module on the vehicle. They are interpreted in the *FrD* (see [section 'Origin Fr1: the Coordinate System \[m\]](#) on page 154 for information about the axis systems). The sensor position is visualized on the graphical representation of the vehicle.

**Rotation X - Y - Z (deg)** These angles specify the rotation of the sensor frame according to the frame on which the sensor is mounted. Rotation order: ZYX.

**Horiz. - Ver. Aperture (deg)** This parameter defines the horizontal and vertical aperture of the sensor beam [deg], maximum angle is 180 deg.

**Longitudinal Range (m)** The longitudinal range of the sensor beam is defined at this point.

**Update (Hz) - Cycles Offset** The parameter Update [Hz] defines the frequency with which the environment is scanned by the line sensor. The Cycles Offset is an option to provide an offset with respect to the number of simulation cycles from the start of the simulation after which the line sensor module should be activated.

**Body Mounting** The mounting point of the *Line Sensor Module* is defined here. By default it is the vehicle based coordinate system *Fr1A*. In case a *Flexible Body* is used you can choose between *Fr1A* and *Fr1B* (see [section 5.2 'CarMaker Coordinate Systems'](#) on page 148 for information about the axis systems and section [section 5.3 'Bodies'](#) on page 150 for more information on flexible bodies).

<b>External Sensor Motion</b>	When this tick box is activated, the external relative sensor travel and rotation is activated, and they can be manipulated by use of the corresponding UAQs <i>LineSensor.&lt;Sensor-Name&gt;.rx_ext</i> , <i>LineSensor.&lt;Sensor-Name&gt;.ry_ext</i> , <i>LineSensor.&lt;Sensor-Name&gt;.rz_ext</i> , <i>LineSensor.&lt;Sensor-Name&gt;.tx_ext</i> , <i>LineSensor.&lt;Sensor-Name&gt;.ty_ext</i> and <i>LineSensor.&lt;Sensor-Name&gt;.tz_ext</i> .
<b>Calculation of relative Position in mounted Frame</b>	Specifies if the line points in the global Line Sensor header file are expressed and sorted relating to the body frame on which the sensor is mounted (e.g.: vehicle frame Fr1A) or the sensor frame.
<b>Reference Point - X- Y- Z (m)</b>	<p>Position of the reference point in vehicle design frame <i>FrD</i>. Coordinates x, y, z [m]. This point is used for the calculation of the relative line points position if relating to the body frame.</p> <p>Default: 0 0 0.</p>

## 5.23.7 Road Property Sensor

The *Road Property Sensor* collects road information at a user defined point ahead of the vehicle along its route. Several attributes like the road bend and its deviation, road marker attributes (speed limit) or the longitudinal and lateral slope are detected. Furthermore, the sensor can provide information about the relative position: deviation distance, deviation angle and current driving lane are calculated. This data can be important for various applications such as lane keeping assistance, lane departure warning, autonomous driving, sign detection, energy management, prescanning, optimization of fuel consumption or detection of wheel liftings. It is possible to define up to 10 sensors of this kind for each vehicle.

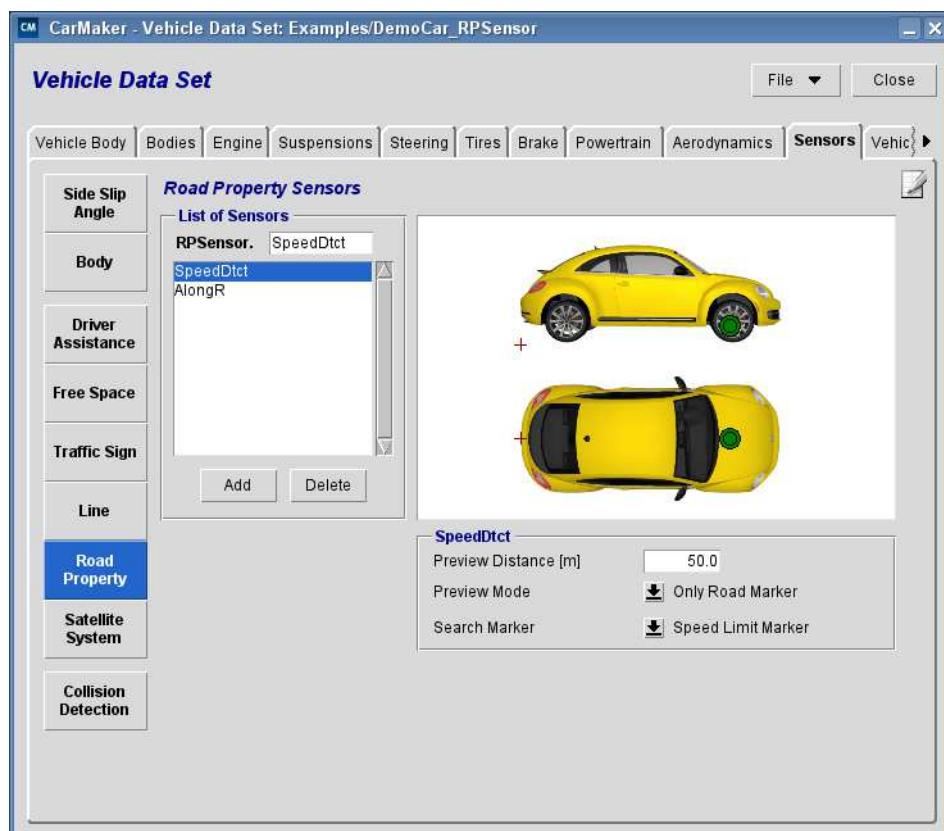


Figure 5.102: Definition of a *Road Property Sensor*

The *Road Property Sensors* are defined by the following parameters.

- RPSensor.** Here you can enter a name for the current sensor, to alleviate the identification of the sensor quantities, e.g. in IPGControl. All User Accessible Quantities of this sensor will start with the prefix *RPSensor.Name.\**. In the example in [Figure 5.98](#) the UAQs of this sensor will start with *RPSensor.RP00.\**.
- Preview Distance (m)** Longitudinal preview distance of the sensor.
- Preview Mode** Several preview modes are available:

Table 5.93: Preview Modes of the *Road Property Sensor*

Mode	Description
Along Route Centerline	Preview along route centerline. The sensor is located at the middle of the front axle. This point is projected in the global xy-plane on the roadway middle line. Starting at the projected point, the previewed road point is reached by following the route centerline with the defined preview distance.
Along Vehicle Direction	Preview along the vehicle direction. The sensor can be placed anywhere on the vehicle. The preview point is placed on a reference axis parallel to the vehicle driving direction vector at the specified preview distance from the mounting point. It is possible to move the preview point to the side or backwards by a rotation angle relative to the reference axis.
Only Road Marker	Only road markers are detected by the sensor, mode is along the centerline.

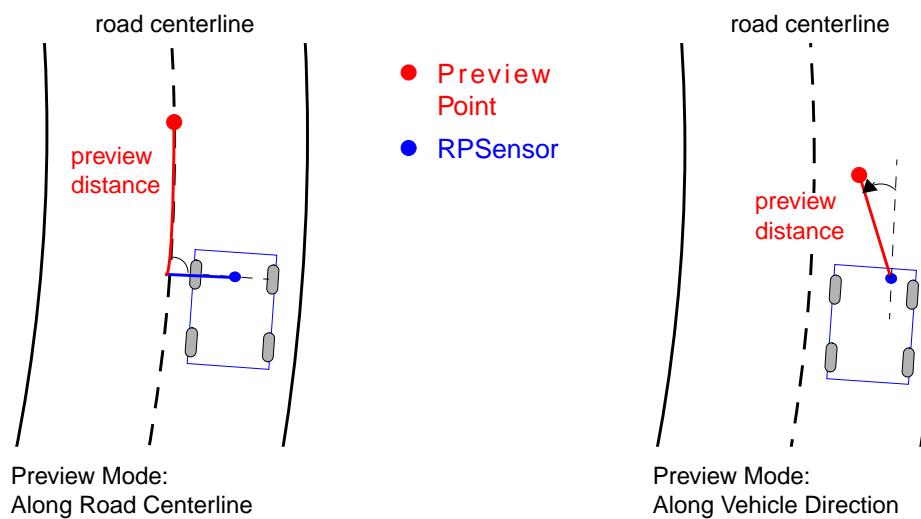


Figure 5.103: Comparison of different *Preview Modes*

- Consider Bumps** This checkbox only is available in the *Preview Modes Along Route Centerline* and *Along Vehicle Distance*. Using this checkbox, the effect of unevennesses on the road (bumps, beams, waves) can be activated/deactivated. The vertical movement of the vehicle is neglected in both cases so that the preview point is always positioned on the road surface.

- Position X-Y-Z (m)** This parameter only is available in the *Preview Mode Along Vehicle Distance*. These coordinates indicate the position of the selected sensor on the vehicle. They are interpreted in the *FrD* (see [section 'Origin Fr1: the Coordinate System \[m\]' on page 154](#) for information about the axis systems). The sensor position can be visualized on the graphical representation of the vehicle.
- Rotation Z (deg)** This parameter is only available in the *Preview Mode Along Vehicle Distance*. This angle specifies the rotation of the preview point around the vertical axis (z-axis) of FrameD in the sensor mounting point on the vehicle.
- Search Marker** This parameter only is available in the *Preview Mode Only Road Marker*. Only the road marker kind defined here is detected by the preview point. Recognized road markers to choose from are the *Speed Limit Marker* or user defined markers. For the latter, the user has to define his own road markers in the *Road dialog*. See [section 4.4.7 'Adding Road Markers' on page 58](#) for more information on the positioning of road markers and the declaration of user markers. As road marker contain attributes, a new quantity called *RPSensor.Name.RMarker.Attr.\** is created for each parameter. Please note, that the *Road Property Sensor* can detect a maximum amount of 10 attributes which are numbers (strings are skipped in the parameter list).



As an example take the *TestRun Examples > CarMakerFunctions > Sensors > RPSensor\_SpeedLimitDetection*.

## 5.23.8 Satellite System

CarMaker offers the possibility to simulate satellites in the orbit and their visibility for the receiver in the vehicle. The so-called *Global Navigation Satellite System* (GNSS) module is activated via the *Environment* module and the *Vehicle Data Set Editor*. In the latter, the receiver position in the vehicle can be parameterized, along with the errors of the signal transmission between satellites and receiver.

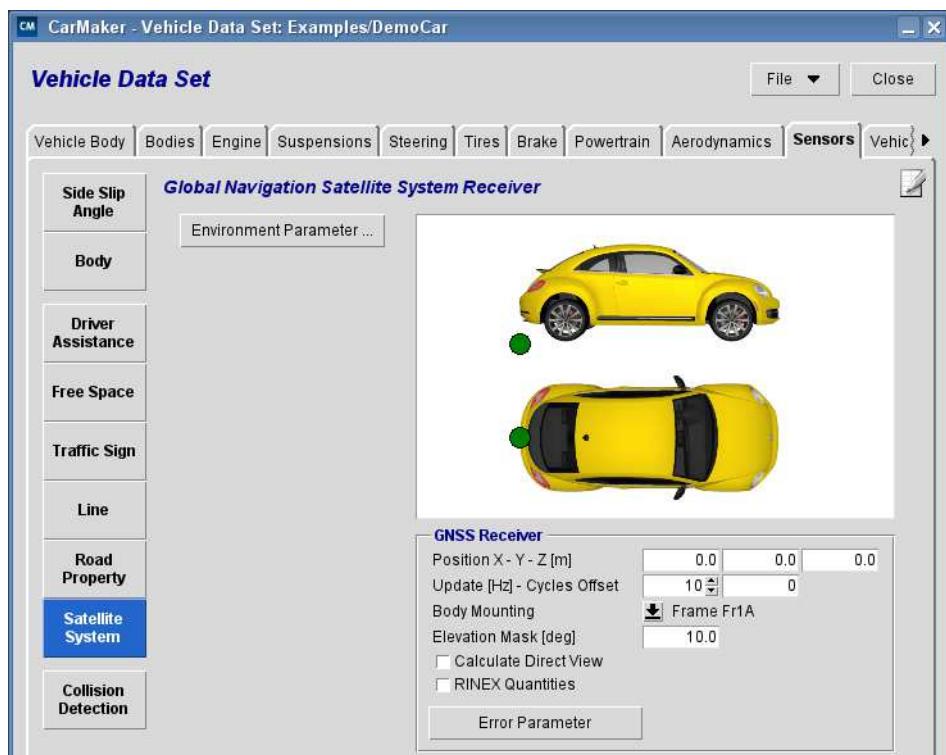


Figure 5.104: Parameters for GNSS module in the vehicle data set

<b>Environment Parameter</b>	This button is a direct link to the Environment dialog, also accessible under <i>Parameters &gt; Environment</i> . The positioning of the satellites is done by the time and date specified in the <i>Environment</i> module. This refers to a navigation message file. Please find further information in <a href="#">section 4.9 'Environment' on page 142</a> .
<b>Position X - Y - Z (m)</b>	These coordinates specify the position of the receiver on the vehicle. They are interpreted in the <i>FrD</i> (see <a href="#">section 'Origin Fr1: the Coordinate System [m]' on page 154</a> for information about the axis systems). The sensor position is visualized on the graphical representation of the vehicle.
<b>Update (Hz) - Cycles Offset</b>	The parameter <i>Update [Hz]</i> defines the frequency with which the environment is scanned and the signals are updated by the sensor. The <i>Cycles Offset</i> is an option to provide an offset with respect to the number of simulation cycles from the start of the simulation after which the line sensor module should be activated.
<b>Body Mounting</b>	The mounting point of the receiver is defined here. By default it is the vehicle based coordinate system <i>Fr1A</i> . In case a <i>Flexible Body</i> is used you can choose between <i>Fr1A</i> and <i>Fr1B</i> (see <a href="#">section 5.2 'CarMaker Coordinate Systems' on page 148</a> for information about the axis systems and section <a href="#">section 5.3 'Bodies' on page 150</a> for more information on flexible bodies).
<b>Elevation Mask (deg)</b>	Only satellites positioned above this aperture angle can be (elevation mask) will be considered in the calculation.
<b>Calculate Direct View</b>	In case, satellites are hidden be traffic objects, their signals will not be transmitted. Without this option, the satellites can always communicate with the receiver, independent of the surroundings, which might block the contact.
<b>RINEX Quantities</b>	Using this option, the parameters from the RINEX file will be available as User Accessible Quantities. Please note, that this option extends the data dictionary by approximately further 700 quantities!
<b>Error Parameter</b>	<p>The signal send by the satellites is usually affected by several error sources. These can be grouped in the following error classes:</p> <ul style="list-style-type: none"> <li>• Receiver Clock Error: This error accounts for the deviation between the clock time of the satellites and the receiver.</li> <li>• Common Mode Error: The common mode errors are caused by the different atmospheric layers, the satellite signal needs to pass. The ionospheric error is caused by the high rate of ionic particles in the upper atmosphere, whereas the tropospheric effects are produced by the pollution of the lowest atmosphere layer.</li> <li>• Receiver Noise: The receiver noise is modeled by a random gaussian algorithm, specified by the standard deviation.</li> </ul>

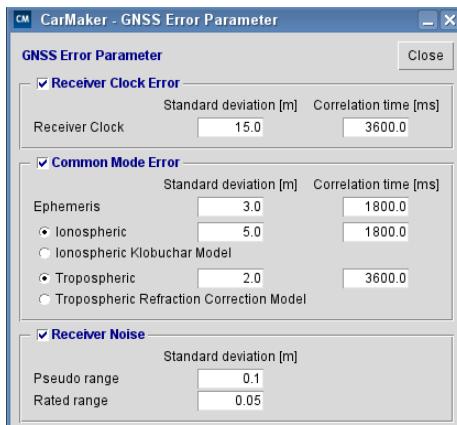


Figure 5.105: Error classes of GNSS module

Please find more detailed information about the GNSS module in the equally named section of the Reference Manual.

### 5.23.9 Collision Detection Sensor

The *Collision Detection* module records if the ego vehicles body or wheels are in touch with an other traffic object. For e.g. to detect collisions with other vehicles during a parking maneuver.

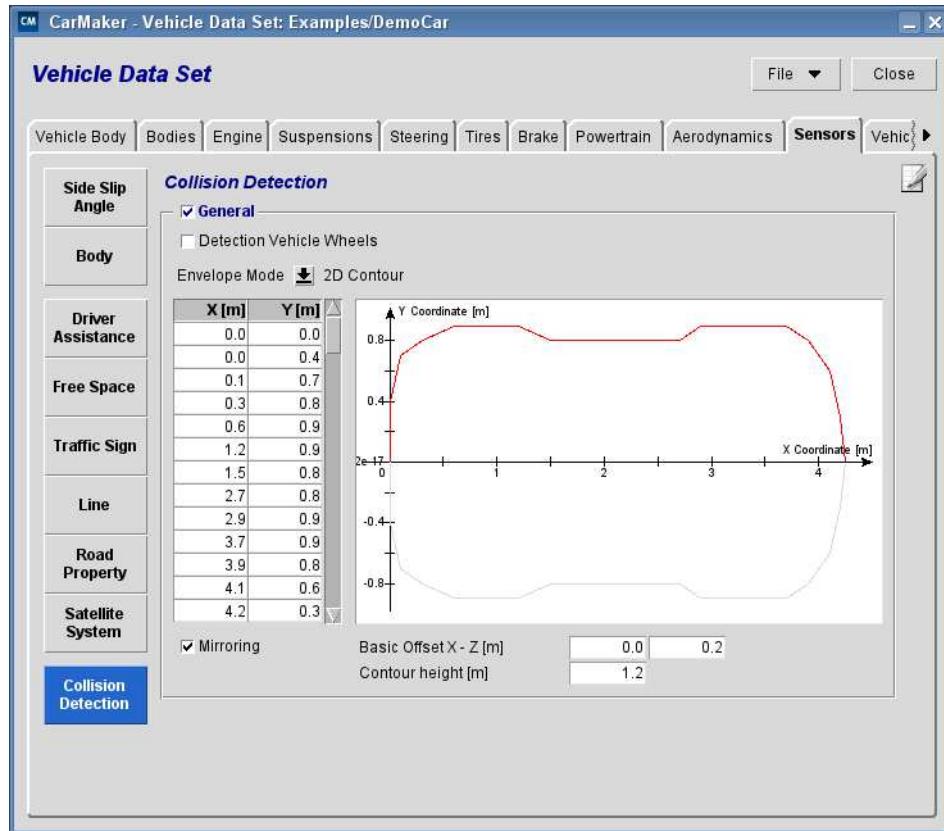


Figure 5.106: Definition of the *Collision Detection* module

The following options are available to configure collision detection.

**General** Select this option to activate collision detection.

**Detection Vehicle Wheels** If this option is selected, the wheels are considered as individual bodies. In this case different User Accessible Quantities are available to distinguish if the vehicle body or the wheel collides with a traffic object.

**Envelope Mode** This parameter defines the outer skin of the ego vehicle seen by the *Collision Detection* module:

Table 5.94: *Envelope Modes* of the *Collision Detection* module

Mode	Description
Cuboid (Outer Skin)	The vehicle's outer skin is a cuboid with the dimensions defined in the vehicle data set under <i>Misc. &gt; Vehicle Outer Skin</i> .
2D Contour	The vehicle's outer skin is defined by a user defined x-y-contour that is extruded in z-direction. It is possible either to define an asymmetric contour or to mirror one side of the vehicle.

**Detection of upper and lower Area**

This check box is available for the envelope mode *Cuboid* only. If it is selected, collisions with the upper and lower surface of the curboid are detected in addition to the four peripheral sides.

For further details about Collision Detection, see the Reference Manual, section "Collision Detection".

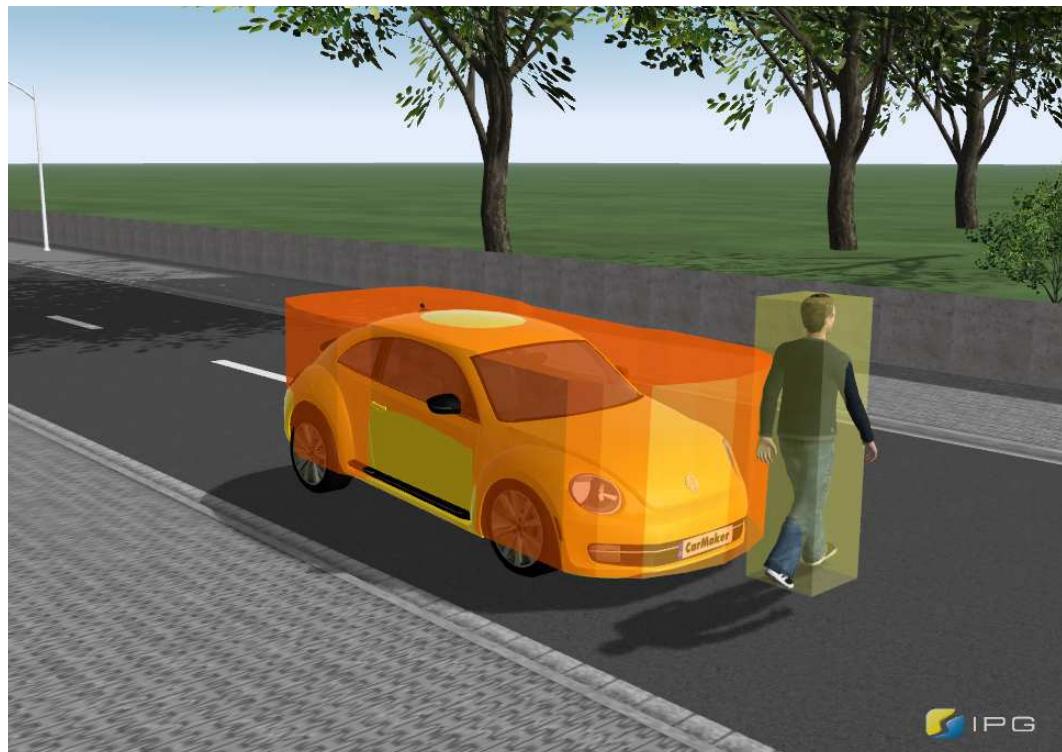


Figure 5.107: Visualization of a collision detected by the Collision Sensor

## 5.24 Vehicle Control

In the Vehicle Control tab, advanced driver assistance systems (e.g. adaptive cruise control, lane keeping assist, emergency braking...) and other global vehicle controller models can be added to the vehicle. They are optional.

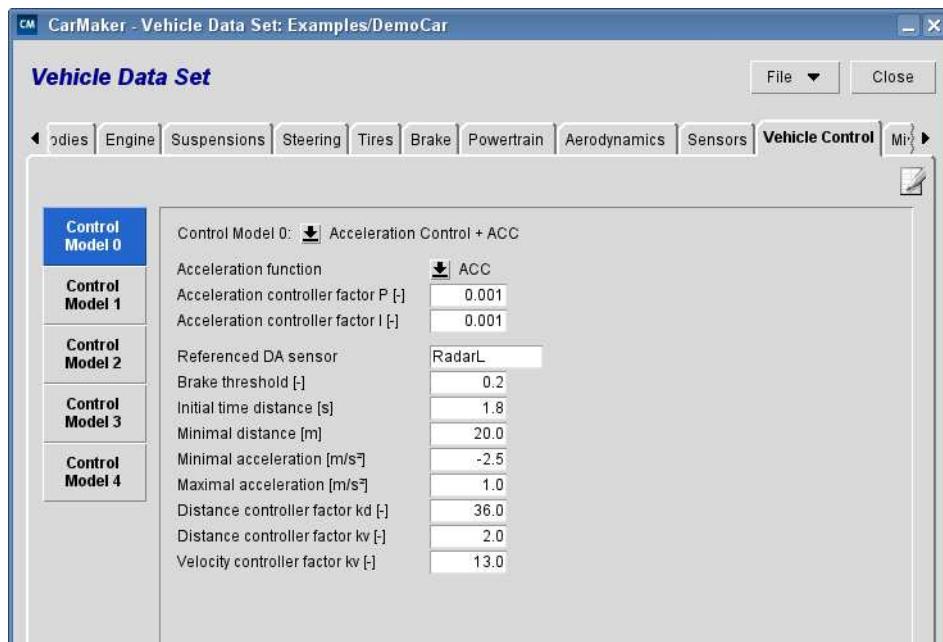


Figure 5.108: Selecting up to five driver assistance systems

If more than one Vehicle Control model is selected, they will be executed in the order *Control Model 0 > Control Model 1 > Control Model 2 > ...*

The following pre-defined examples are available:

Table 5.95: Pre-defined Vehicle Control models

Name	Description	Example Vehicle
Acceleration Control + ACC	Includes and advanced driver assistance system with acceleration control	Demo_BMW_5
GenLongCtrl	Generic longitudinal controller with simplified AEB and FCW functions	DemoCar_EuroNCAP
GenLatCtrl	Generic lateral controller with simplified LKAS and LDW functions	DemoCar_EuroNCAP

Please find further information about the models and their parameters in the [Reference Manual, chapter "Vehicle Control"](#).

## 5.25 Miscellaneous

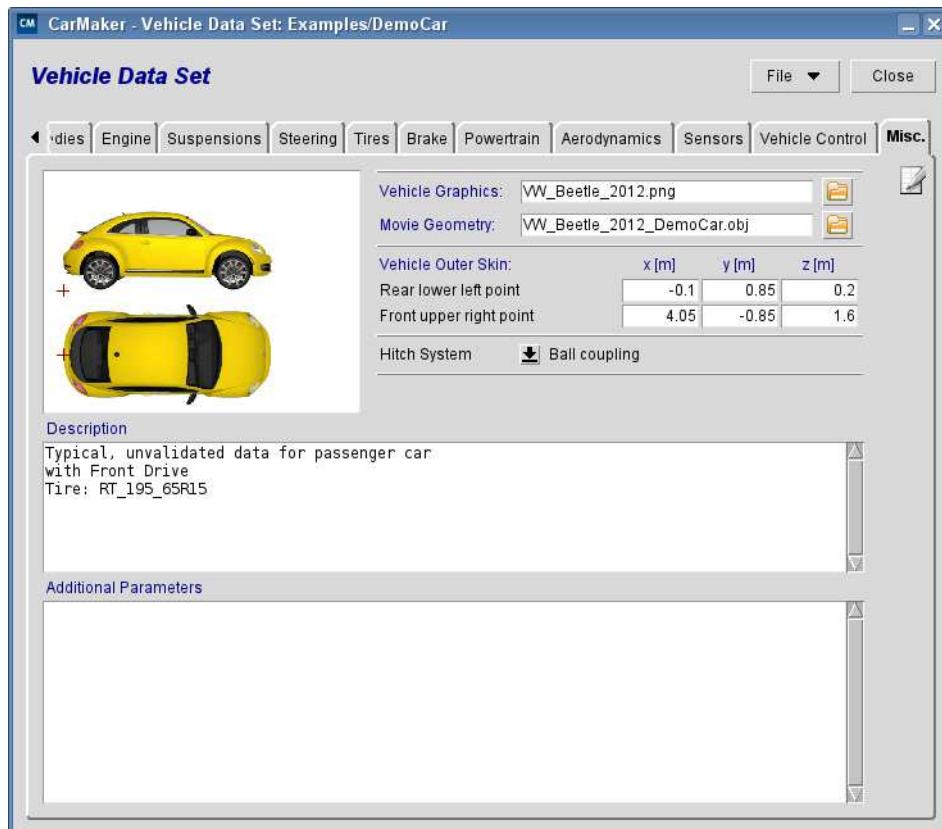


Figure 5.109: A few additional parameters can be defined.

### 5.25.1 Vehicle Graphics

By clicking on *Select*, you can select the picture that is displayed in the vehicle editor. A set of examples is already available (e.g. the New Beetle).

You also can create your own vehicle graphics. Supported file types are PNG images or a tcl/tk script. Using PNG pictures, any photo in the PNG format can be imported. A picture of the car in top view and another one in side view is required.

In the following it will be described how an PNG image can be created using IPGMovie. First, two screenshots need to be taken. For this, the following IPGMovie settings are recommended:

- Camera > Bird's Eye View / Right Side in IPGMovie
- field of view = 10 (Camera > Camera Settings)
- maximize IPGMovie window
- camera placed in the middle of the screen

- zooming out with fixed camera position (View > Fix Camera View Direction)



Figure 5.110: Screenshots of the top view and side view of the car in IPGMovie

When this is done, the screenshots have to be reworked for the display in CarMaker. The screenshots must be loaded into a graphic tool. Cut out the car along its outer shape and paste it to a new file, aligned with its back at the left margin of the window (in case of a side view additionally align it vertically to the bottom). Then, the picture needs to be scaled to the length of the car (1m=100pixel) and saved as a PNG image. Using the graphic tool, everything around the car needs to be deleted to create a transparent background.

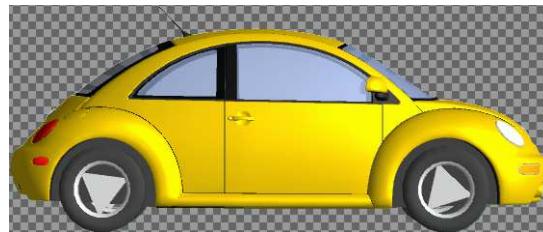


Figure 5.111: Car with transparent background

Now select all and paste it to another file of the size of 560x226 pixel (72 dpi). Again move the picture to the left side of the window (in case of a side view additionally align it vertically to the bottom). To export the picture, the side view needs to be saved as "filename.png" to the *Data/Pic* directory of your project folder. The top view should be saved to the same folder as "filename.top.png". Finally, the pictures can be selected in the CarMaker GUI

Finally, to display your car in the CarMaker main GUI and in the vehicle data set window, go to *Parameters > Car > Misc* and select your picture (side view) in the field *Vehicle Graphics*.

To understand how the tcl/tk script works, you can edit one of the examples (Wordpad recommended) to look at the syntax and then create your own graphic. The files corresponding to the vehicle graphics are located in the folder *./Data/Pic* of a working directory.

## 5.25.2 Movie Geometry

At this point you can select the so called "movie object" which is used by IPGMovie for the animation. This movie object is used only for the vehicle being simulated, not for the traffic (for this, read the [section 4.8.2 'Defining Traffic Objects' on page 128](#)). If you do not choose any movie object here, IPGMovie will use the ABRAXAS object by default. More information about the ABRAXAS object can be found in the [section 7.1.1 'Navigating with the camera' on page 398](#).

An example of a movie object corresponding to the New Beetle is available, but you can generate your own movie object using your preferred CAD software.



Alternatively you can buy a movie object on specialized websites. Search on the web with the following key words: "3d model download 3ds car Wavefront vehicle graphic obj".

In any case please read the [section 7.4 'IPGMovie Object Files' on page 428](#), you will find all necessary information you need to generate or choose your movie object.

**Elastically mounted parts****TruckMaker: Drivers Cab and Platform**

Using TruckMaker, you are able to simulate an elastically mounted drivers cab platform with two additional degrees of freedom. The cab and platform models are automatically loaded if the object files are named as follows:

- Chassis model: *Truck.obj*
- Driver's cab: *Truck.Cab.obj*
- Platform: *Truck.Plf.obj*

**MotorcycleMaker: Swing Arm and Fork**

Using MotorcycleMaker, you are able to simulate a swing arm and a fork with two additional degrees of freedom. The swing arm and fork models are automatically loaded if the object files are named as follows:

- Chassis model: *Motorcycle.obj*
- Swing arm: *Motorcycle.SFB.obj*
- Fork: *Motorcycle.SFT.obj*

### 5.25.3 Others

**Vehicle Outer Skin (m)**

At this point the size of the ABRAHAS object displayed in IPGMovie can be defined. More information about the ABRAHAS object can be found in the [section 7.1.1 'Navigating with the camera' on page 398](#).

**Hitch System**

Select the hitch system for the trailer coupling.

**Description**

Here you can write some free comments. These comments are displayed at the bottom of the selection window when you load a vehicle data set.

**Additional Parameters**

If you have developed a special model, you may need to parameterize it. In this case, you can write some additional parameters in this field that will be written in the vehicle file, and then read the additional parameters directly from the vehicle file.

Of course, this field is only useful for a power user who wants to extend the CarMaker models.

## 5.26 Trailer Model

The following subsections describe all parameters required to parameterize the trailer model using the *Trailer Editor*. It is reachable in the CarMaker GUI by clicking on Parameters > Trailer. The technical background to each model can be found in the Reference Manual.

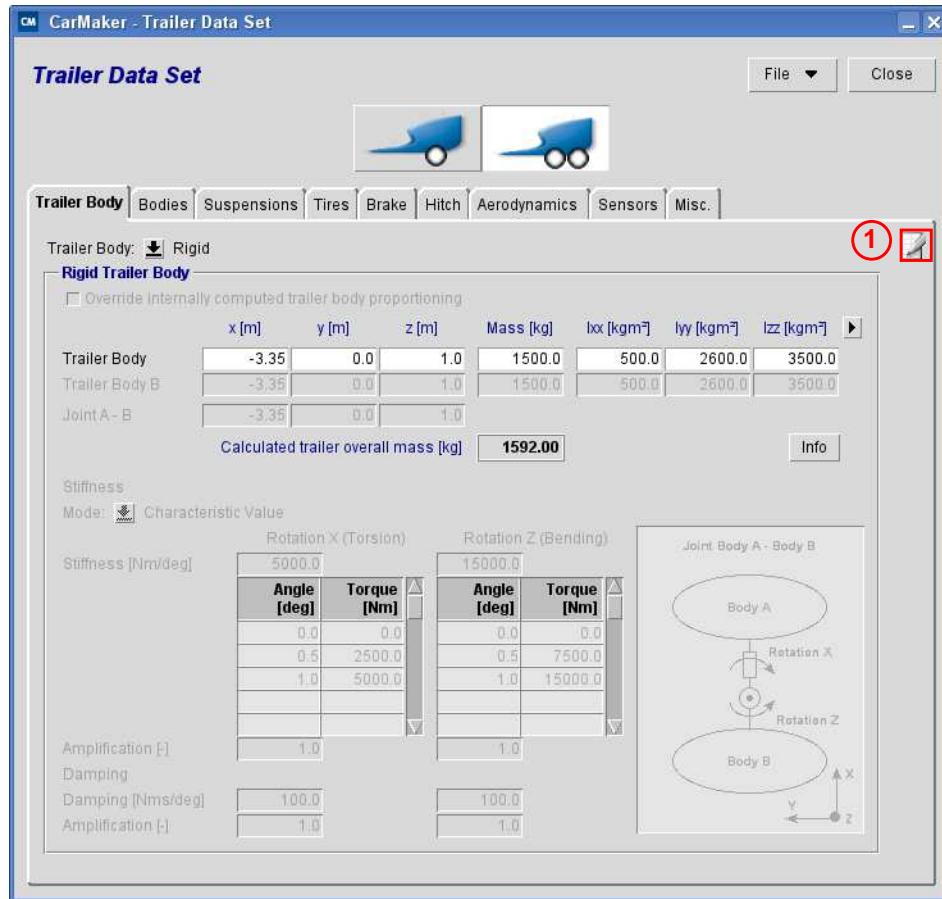


Figure 5.112: The Trailer Editor

### Tips for the Trailer Editor



Please note, that you can add some comments in each tab. For this, click on the icon in the top right hand corner of the tab (see [Figure 5.112: The Trailer Editor](#) box 1).



Figure 5.113: Trailer Comment



In the "Excel-like" tables (e.g. Aerodynamics tab or spring stiffness): by clicking only once in a cell, you can move to another cell by using the keyboard arrows. If you double-click, you can move to another digit within the cell by using the keyboard arrows and thus edit the cell's entry.

## Managing the Trailer Data Set Library

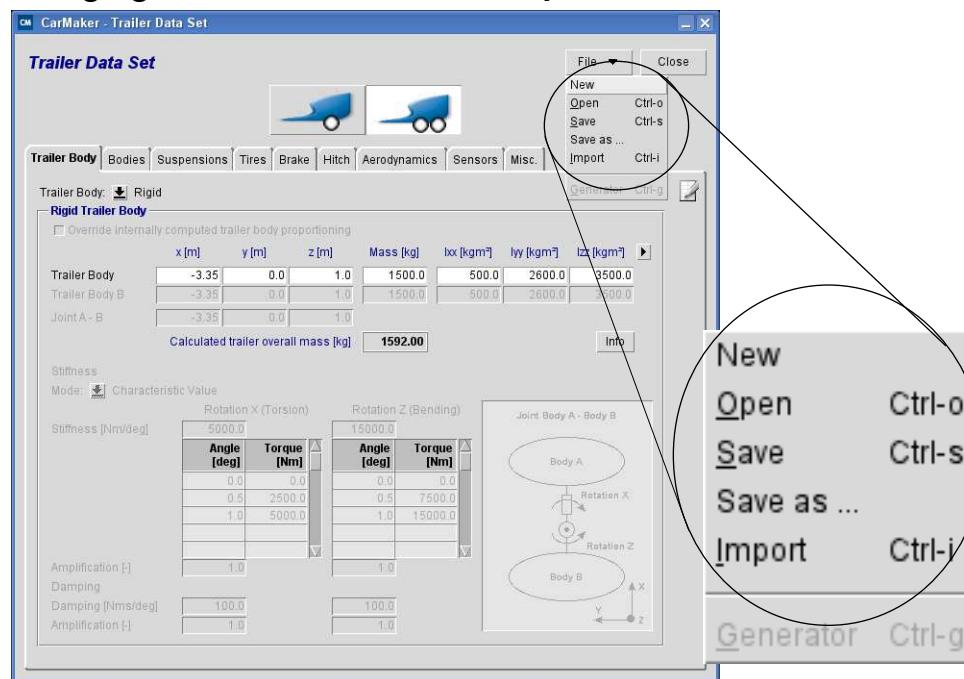


Figure 5.114: Trailer Data Set Manager

However, below you can find a short sum up of the functionalities that are available:

- To use a data set which is already built, you just have to load the corresponding trailer file via the trailer editor. Click on *File > Open*.
- You can also create a new trailer data set from scratch. In this case, be aware that even if you choose this option, the trailer editor is filled with default values (at the exception of the trailer picture). If you did not finish the trailer parameterization before closing, do not forget to add a comment where you stopped (field *Description* in the tab *Misc.*). Click on *File > New*.
- Once you have parameterized your trailer data set, you can save your changes or alternatively save them to a new file. Click on *File > Save* or click on *File > Save As*.
- Note that you have the possibility to import a part of the trailer from another trailer data set. Click on *File > Import*.

### 5.26.1 CarMaker Coordinate Systems

In CarMaker, coordinate systems - called frames - with different origins are defined. However, the direction of the axes is always the same, following DIN 70000:

- x-axis: Points in driving direction of the trailer
- y-axis: Points 90 deg to the left side of the x-axis
- z-axis: Points upwards vertically

An overview of all frames available can be found in the Reference Manual, chapter 1.2 "Car-Maker Axis Systems". The reference frame for parameterization of the trailer data set is *Fr1 (Frame One)*. Its origin is located at the foremost point of the trailer, on the ground.

## 5.26.2 Bodies

The masses of the trailer models are described in the tabs *Trailer Body* and *Bodies*

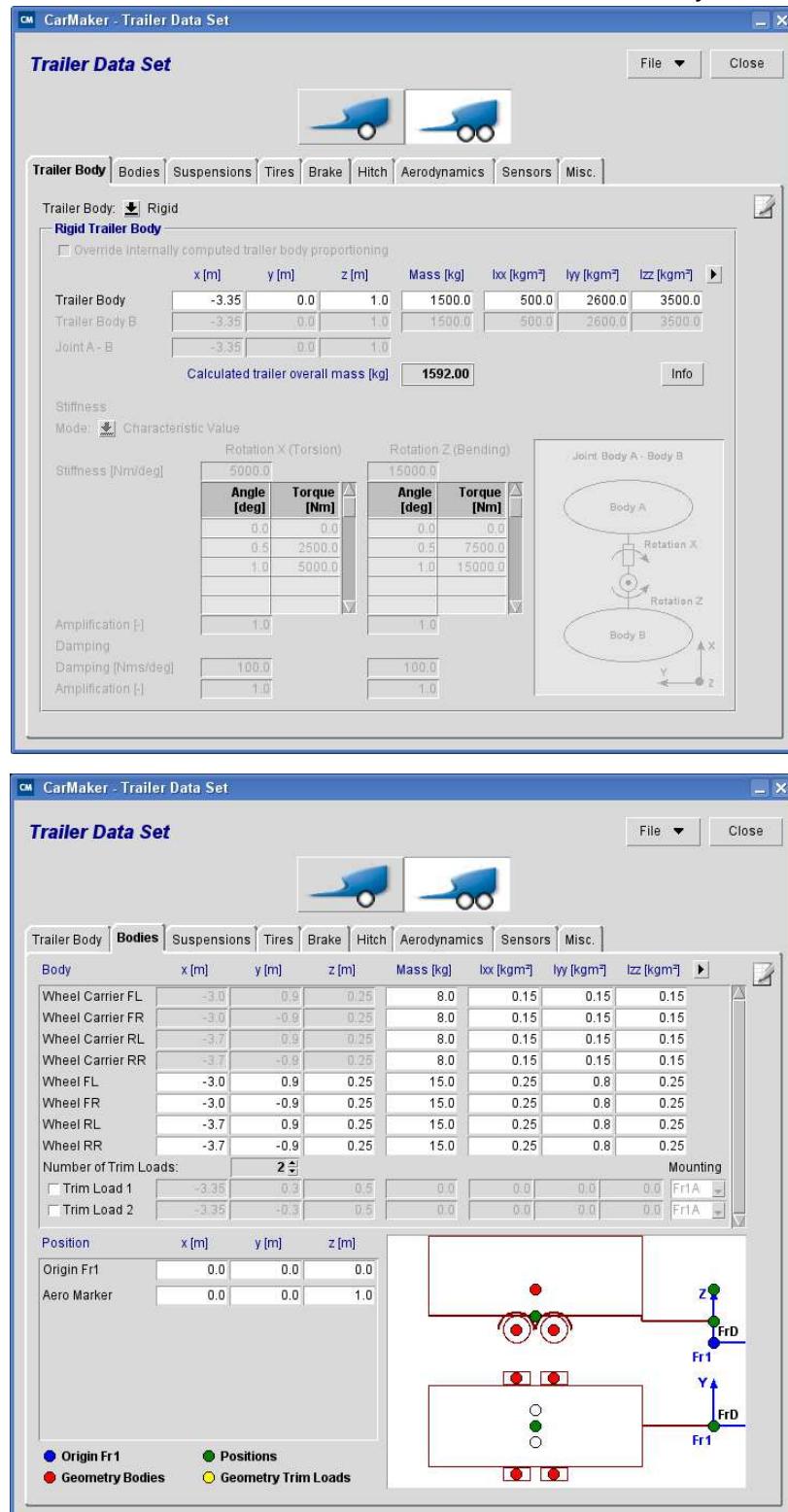


Figure 5.115: Definition of the main body and sub bodies in the Trailer Data Set editor

## Mass and Inertia Definitions

A different number of bodies can be parameterized. The unsprung masses are defined in the *Bodies* tab under *Wheel* and *Wheel Carrier*. Heavy loads can be extracted of the main body and defined separately using trim loads.

Note: All masses of the *Bodies* tab are added to the main body's mass internally.



The following tables give an overview of the masses available:

Table 5.96: Definition of the main bodies in the *Trailer Body* tab

Name of the Body	Comments	Tab
Trailer Body A	Includes all sprung masses, optionally with the exception of user defined trim loads.	Trailer Body

Table 5.97: Definition of the extra masses in the *Bodies* tab

Name of the Body	Comments	Tab
Wheel Carrier	Unsprung but non-rotating masses (wheel carriers, brake callipers, rods or other fixed elements whose weight pushes on the wheel carrier). If an element is only half unsprung, then only the half of its mass should be regarded here.	Bodies
Wheel	Unsprung and spinning masses (wheels, disk brakes). The position of the wheels is defined assuming that the tires are not compressed.	Bodies
Trim Loads	Optional to add numerous extra loads.	Bodies

x, y, z (m)

Mass (kg)

Ixx, Iyy, Izz (kgm<sup>2</sup>)

Each of the masses has the following properties:

- Position of the center of gravity of the main body in the three fields x, y and z of the *Trailer Body* tab.  
The location of each body, especially of the main body, is parameterized in the design state. The design state can be a virtual state (possibly even be a state where the springs are not compressed) or the static state. The position of the wheels is defined assuming that the tires are not compressed either.  
Note that the state in which you parameterize the positions of the masses has an influence on the parameterization of the kinematics.
- Under *Bodies* define the weight influence of the body.
- Under *Ixx*, *Iyy* and *Izz* the moments of inertia of this body mass can be defined in the axis orientation of frame1. The position of the center of gravity of the mass regarded is taken as origin.
- By clicking on the small arrow that is at the end of the line (near *Iyy [kg\*m<sup>2</sup>]*), you have access to the products on inertia *Iyz*, *Ixz* an *Ixy*. The products of inertia are of lower importance, they can be left empty in a first approximation.

Overall mass

The field calculated overall mass in the main body tab states the current weight of the whole trailer including all masses defined separately in the *Bodies* tab.

## Other Parameters

x, y, z  
(m)

In the tab *Bodies*.

- Aero Marker*: This parameter belongs to the aerodynamic parameterization. See the section 5.26.7 'Aerodynamics' on page 325.

- *Picture:*

By bringing the cursor on the colored points symbolizing the position of the bodies, you can display the coordinates of those bodies.

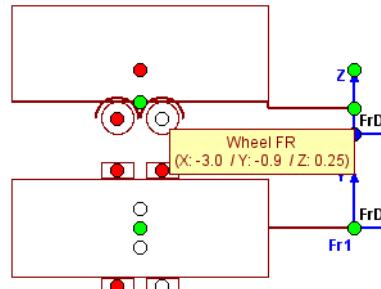


Figure 5.116: Body Coordinates

The picture that is displayed in the trailer editor can be chosen in the tab *Misc.* of the trailer data set editor. You can define your own picture, too. Please refer to [section 5.26.9 'Miscellaneous' on page 332](#).

### 5.26.3 Loads and Trim Loads

Let us give a few information about the additional loads.

As you have seen, with the so called *Trim Loads* there is the possibility to add some extra loads in order to simulate passengers, luggage, heavy measurement equipment and so on. If you check the static position of the trailer using the ModelCheck, the effects of trim loads will be included e.g. to the height of the trailer center of gravity or the wheel positions in the start off configuration.

For some specific simulation purposes, you may want to simulate a TestRun with an additional load on the trailer, but you still want to check the static state of the trailer without it. This is also possible with the so called *Loads* (not to be confused with the *Trim Loads!*). The *Loads* are saved in the TestRun file, whereas the *Trim Loads* are saved in the trailer data set. *Loads* can be specified in the CarMaker main GUI, under *Parameters > Loads*:

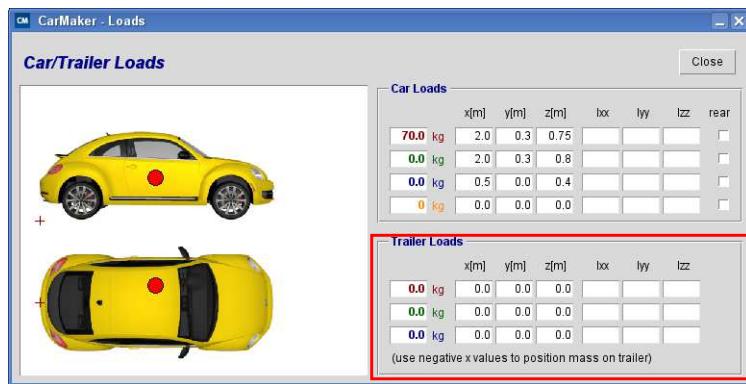


Figure 5.117: Definition of Loads

The number of *Loads* is limited to 4 for the car, and 3 for the trailer. The *Loads* are not considered during a ModelCheck analysis. But you may ask: How should I decide to use the *Loads* or the *Trim Loads*? Basically, you should use the *Trim Loads*. Indeed the number of *Trim Loads* is unlimited, and you can parameterize them in the same window as the other bodies. The *Loads* are interesting if you would like to simulate a TestRun with a load, but check the static state without it.

## 5.26.4 Suspension

### Spring

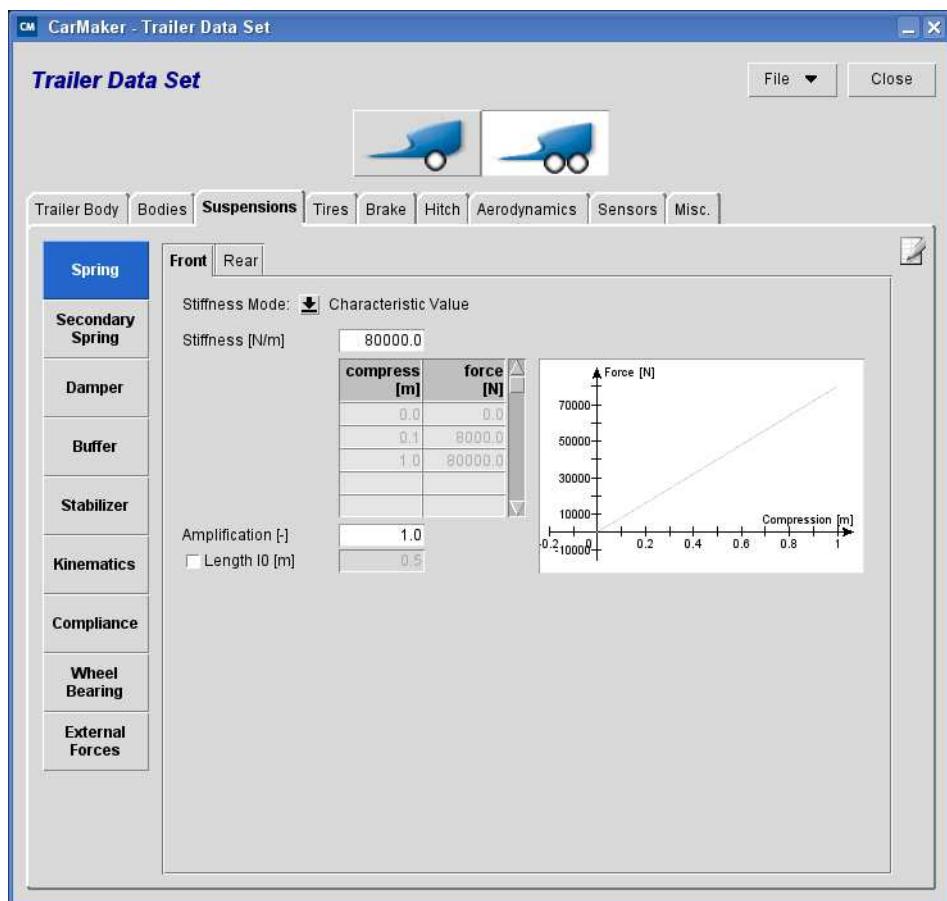


Figure 5.118: Definition of the spring characteristics

For a given axle, the springs characteristics are the same for both sides of the trailer. The spring is modeled as a component that generates a force when it is compressed / stretched. The spring is seen here as a "stand alone" component which properties (free length and stiffness) can be measured independently.

This means that you do not have to specify the geometrical properties. Its length variation in function of the wheel travel is defined in the kinematics section (see [section 'Suspension Kinematics' on page 313](#))

For additional information, also see the Reference Manual section 'Springs'.

**Mode** For each axle, the spring stiffness can be defined either with a simple coefficient (mode *Characteristic Value*) if the spring force is a linear function of the spring length variation or in a table (mode *Look-Up Table 1D*) otherwise.

**Stiffness (N/m)** This parameter either is a simple coefficient or a table according to the selected parameter *Kind*. A positive spring length variation means that the spring is compressed.

If the application requires it, you may also parameterize the stiffness for the traction domain (negative length variation). If undefined, CarMaker estimates the characteristics in this domain by using the first value given and keeping it constant.

**Amplification (-)** This parameter enables to scale the spring stiffness very quickly for test purposes instead of modifying the whole table. You can also use this parameter to convert the values to fit to the units required by CarMaker. The default value is 1.0.

**Length  $l_0$  (m)** This is the relaxed or unstretched length of the spring. The resulting spring force depends on the difference between the relaxed length  $l_0$  and the length. The length is the current distance between the lower and upper attachment point of the spring. Deactivating the checkbox lets CarMaker calculate the length of the unstretched spring.



Before starting a simulation with a new trailer data set, you should check if the value for the relaxed spring length is correct, or if it needs to be tuned. For this, you can compare the parameter *spring coord* in the equilibrium state output of the Model Check. Please refer to the section 6.4 'Model Check' on page 360.

## Secondary Spring

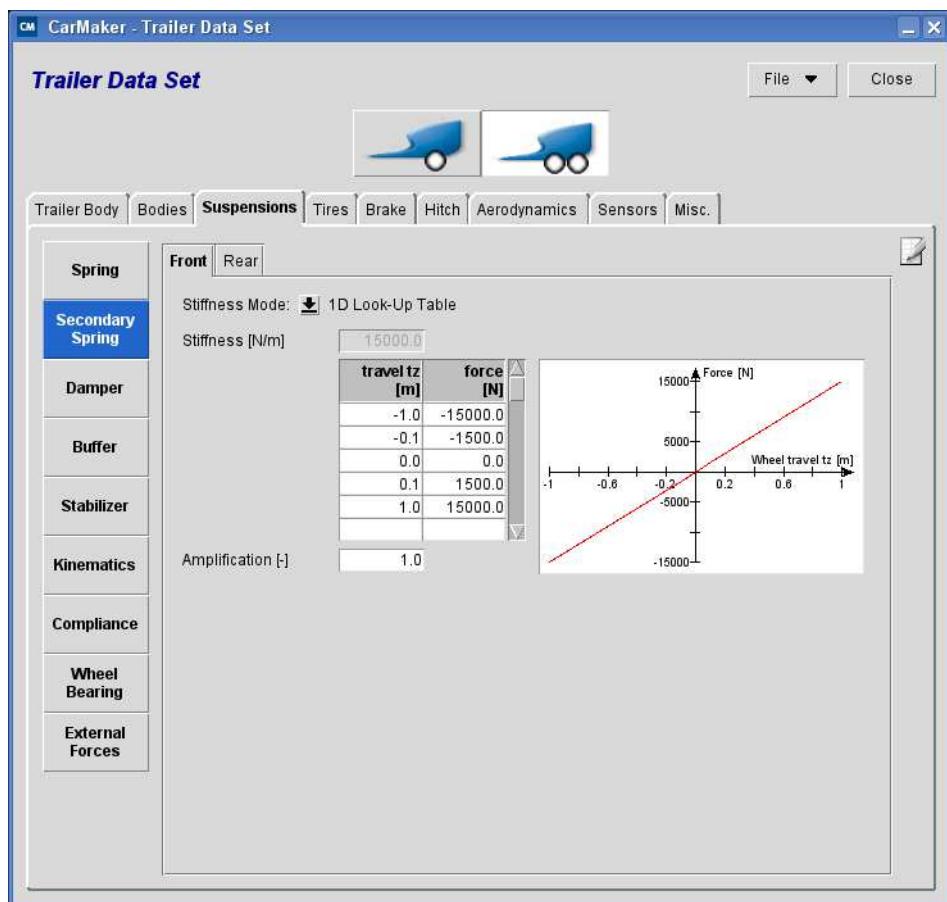


Figure 5.119: Definition of the secondary spring characteristics

**Mode** For each axle, the stiffness of the secondary spring can either be defined with a simple coefficient (mode *Characteristic Value*) if the secondary spring force is a linear function of the wheel travel variation, or in a table (mode *Look-Up Table 1D*) otherwise. If the mode *-not specified-* is selected, the secondary spring is deactivated.

**Stiffness (N/m)** This parameter either is a simple coefficient or a table according to the selected parameter *Mode*. A positive wheel travel  $tz$  means that the wheel center is lifted up and that it generates a positive secondary spring force.

- Amplification (-)** This parameter enables to scale the stiffness of the secondary spring very quickly for test purposes, instead of modifying the whole table. You can also use this parameter to convert the values to fit to the units required by CarMaker. The default value is 1.0. If the value is 0.0, the secondary spring is deactivated.

## Damper

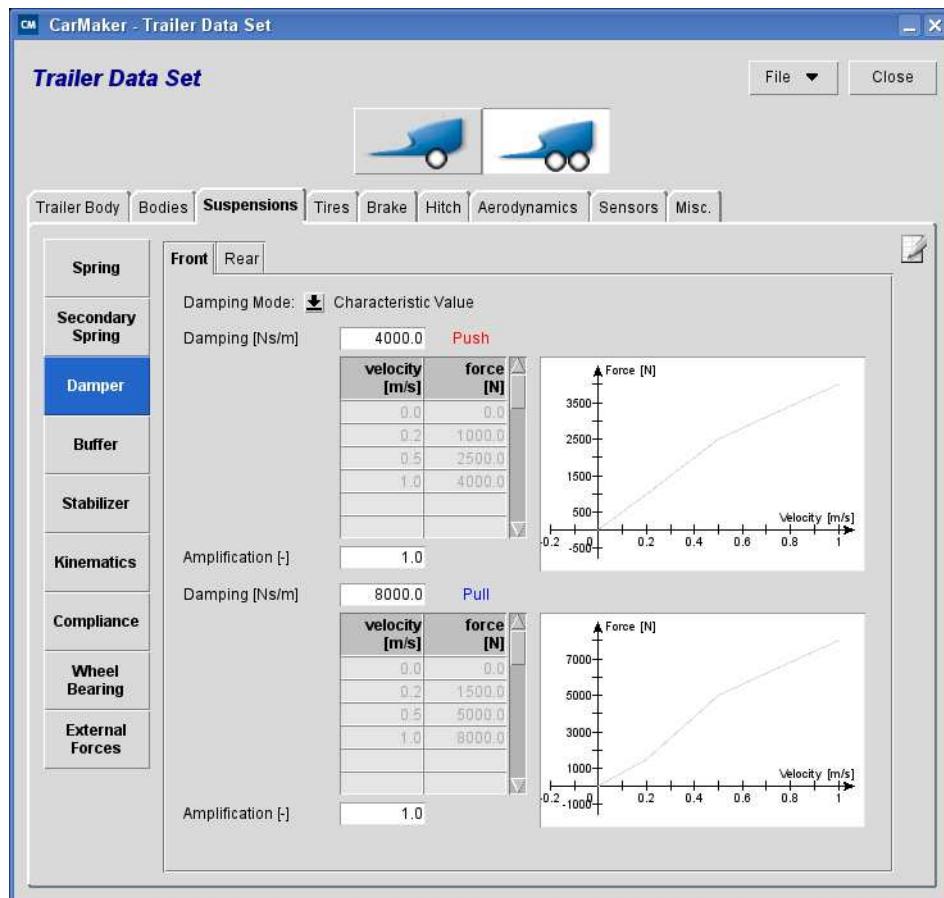


Figure 5.120: Definition of the damper characteristics

For a given axle, the damper characteristics are the same for both sides of the trailer. The damper is modeled as a component that generates a force when being compressed / deflected (reaction to the change of velocity). Its length variation in function of the wheel travel is defined in the kinematics (see [section 'Suspension Kinematics' on page 313](#)).

For additional information, see the Reference Manual section 'Dampers'.

- Mode** Either the damper characteristic is specified by a single coefficient (mode *Characteristic Value*) or by a table (mode *Look-Up Table 1D*).
- Damping ( $N^*s/m$ )** For each axle, the damper characteristic is defined either by a coefficient or by a table of values according to the selected mode. The characteristics of the damper are split into two domains: push and pull. For each domain you can define a different damping. The push domain corresponds to a positive damper speed, which means that the damper is compressed.
- Amplification (-)** This parameter enables to scale the damper characteristics very quickly instead of modifying the whole table. You can also use this parameter to convert the values entered to the units required by CarMaker.

## Buffer

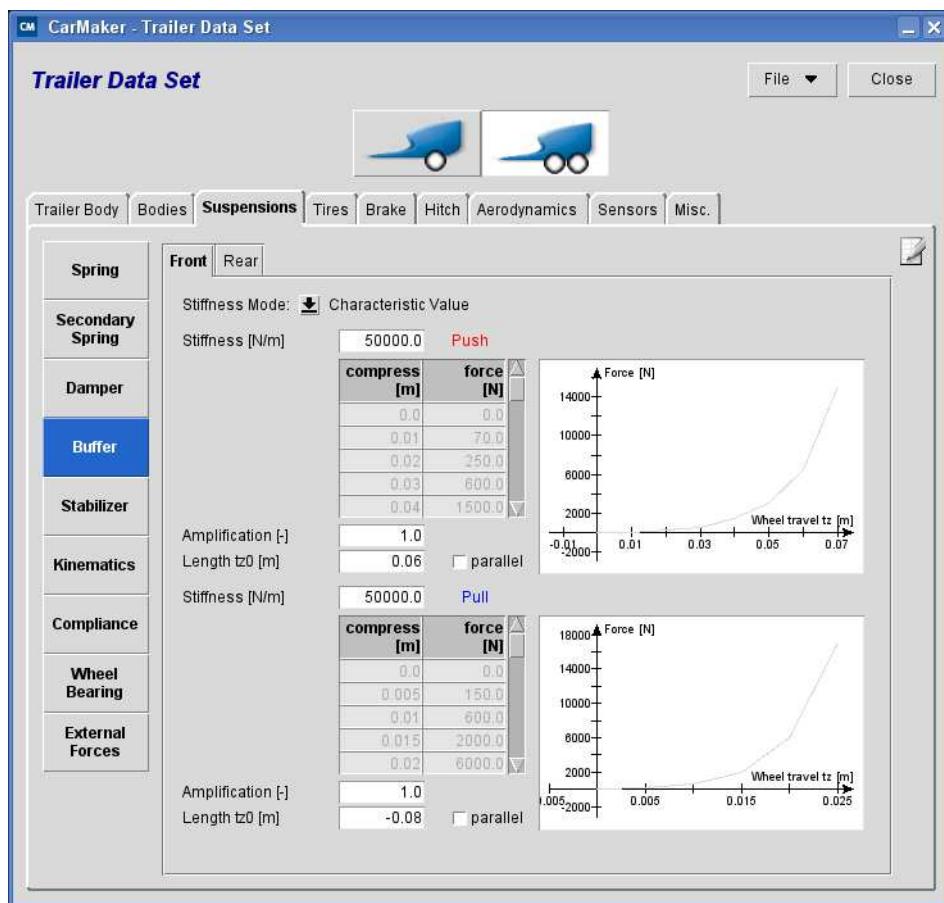


Figure 5.121: Definition of the buffer characteristics

For a given axle, the buffer characteristics are the same for both sides of the trailer. The buffer is modeled as a spring which is activated only at a defined wheel travel. Its length variation when activated is defined in the kinematics (see [section 'Suspension Kinematics' on page 313](#)).

For additional information, see the Reference Manual section 'Buffers / Bumpers'.

The buffers are used to limit the wheel travel in one direction or in both up and down. If the wheel travelled far enough to hit one of the buffers (for instance the push buffer), the buffer acts like an additional spring.

That is why you have to parameterize two buffers per axle, a stiffness for each buffer, and the wheel travel from which the buffers are activated for both positive and negative direction.



Note that you may parameterize the buffer influence directly by the spring stiffness. If you have already added the buffer stiffness to the spring stiffness, you do not need the buffer model any longer. In that case you just need to deactivate the real buffer model by setting the amplification factor to "0" (see below).

**Mode** The buffer stiffness can be parameterized by a coefficient (mode *Characteristic Value*) or by a table of values (mode *Look-Up Table 1D*).

**Buffer Stiffness (N/m)** For each front and rear axle, the damper characteristic is defined either by a coefficient or by a table of values according to the selected mode.  
The characteristics of the buffer are split into two domains: push and pull, where the push domain corresponds to a positive wheel travel.

- Amplification (-)** This parameter enables to scale the buffer stiffness very quickly instead of modifying the whole table. You can also use this parameter to convert the values entered to the units required by CarMaker.
- Length tz0 (m)** This length is the wheel travel in the vertical direction from which the buffer is activated ( $z$  axis of the frame  $Fr1$ ), calculated from the position in the static equilibrium configuration (see [Figure 5.122: Buffer elements at the left wheel](#)).
- tz0 parallel** If this flag is activated, the compression  $tz0$  is for parallel compression. Without this option,  $tz0$  is calculated by varying only one wheel travel whereas the opposite wheel is kept uncompressed. .

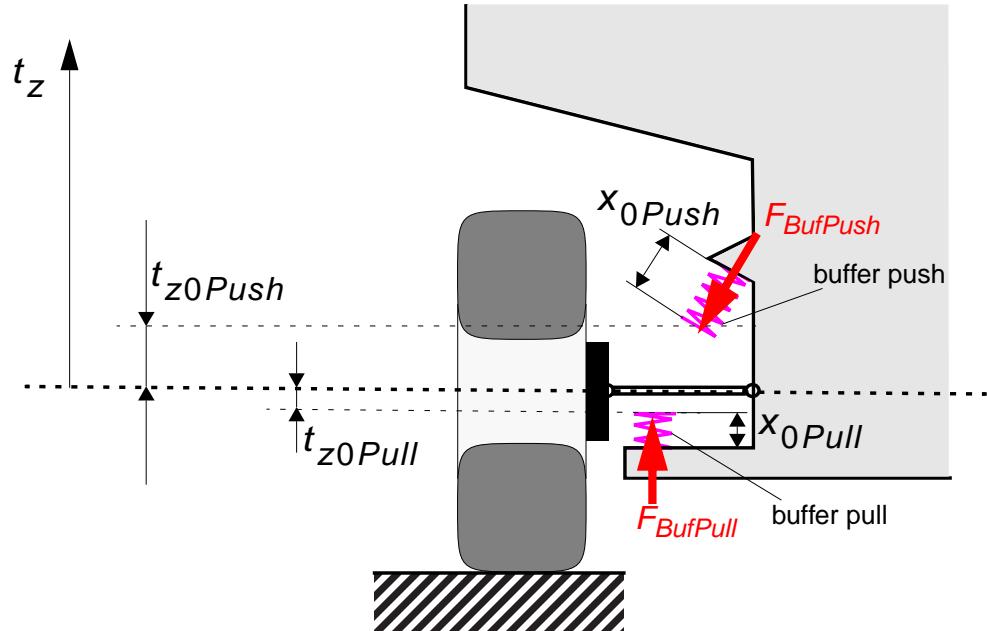


Figure 5.122: Buffer elements at the left wheel

## Anti Roll Bar (ARB)

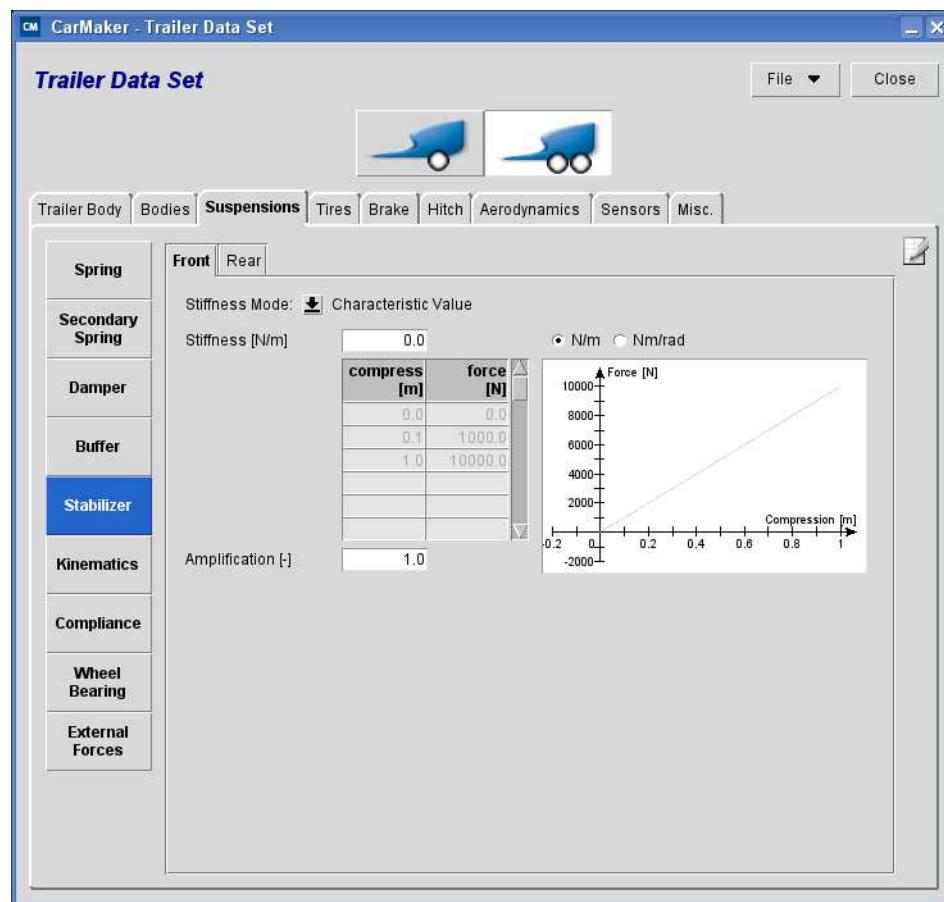


Figure 5.123: Definition of the anti roll bar characteristics

At this point, only the properties of the single ARB element are specified. Its angle or length variation is defined in the kinematics (see [section 'Suspension Kinematics' on page 313](#)).

For additional information, see the Reference Manual section 10.5 'Suspension Roll Stabilizer / Anti-Roll Bar'.

### Stiffness Mode

**Stiffness  
(N/m)  
or (Nm/deg)**

This is the stiffness of the anti roll bar. Two options are available:

- the ARB as a virtual spring: unit of the stiffness is N/m.  
In that case, the displacement typically considered is the movement of the ARB end rod. This displacement should be very small.  
Attention: the stiffness is dependent on the place where the force is applied. Please find further information in the Reference Manual.
- the ARB is a real torsional component: the unit of the stiffness is Nm/rad.

**Amplification  
(-)**

This parameter enables to scale the ARB stiffness very quickly instead of modifying the parameter itself. You can also use this parameter to convert the values entered to the units required by CarMaker.

## Suspension Kinematics

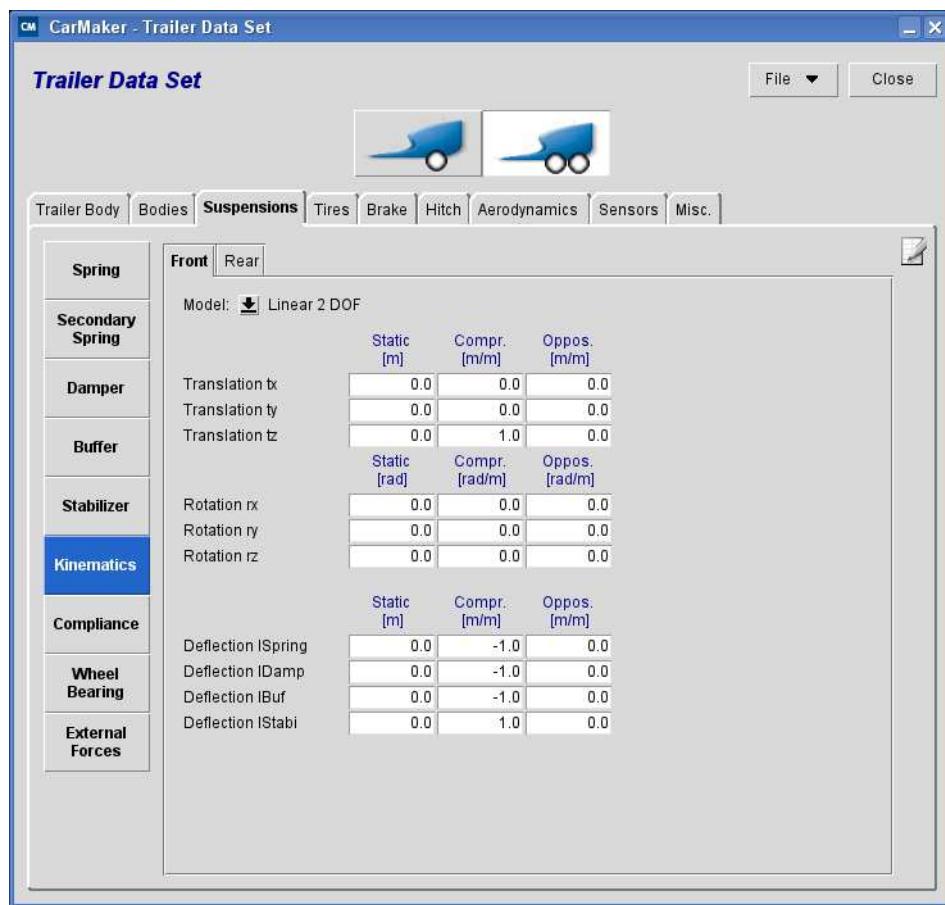


Figure 5.124: Definition of the kinematics

### Concept of the Kinematics Model

Basically, CarMaker ignores the current geometry of the axle. CarMaker knows the original position of the wheel (parameterized in the tabs *Bodies* and *Trailer Body*), and the variations of these positions according to the wheel travel. The position variations are described precisely by the kinematics model. For several states of wheel compression, the resulting position of the wheel center is defined by its translation in x, y, z and rotation around the x, y, z axes in relation to the static state.

In addition to the calculation of the wheel position, the kinematics model also includes the length variation of the spring, damper, buffer and ARB against the same variables.

Knowing these kinematics tables, CarMaker can calculate the forces applied to the wheel center, for any independent axle or even for a rigid rear axle (see the description of the *Model*).

For additional information about the kinematics model, see the Reference Manual section 11.1 'Overview' and particularly the section 'Kinematics Models'. For additional information about the internal degree of freedom, read the Reference Manual section 11.1.1 'Describing Kinematics with Generalized Coordinates'.

### Description of the Kinematics Models

Various kinematics descriptions are available. The kinematics have to be specified for each axle separately. Usually, a suspension is symmetrical, but a non-symmetric axle can also be parameterized using the mode *External File*.

**Model** In general, two categories of kinematics characteristics are available which can be selected under *Model*. There are different linear kinematics descriptions with one, two or three degrees of freedom and the *External File* model for non-linear characteristics:

Table 5.98: Kinematics Models

Mode	Description
Sleeve Axle	Simplest model of a suspension. It is characterized through a straight translation of the wheel carrier. There is no modification of toe or camber whilst compression.
Crank Axle	Most commonly used suspension for passenger car trailers. It has a rotating axis of the crank arms perpendicular to the trailers roll axis. The wheel carriers are connected at the end of the crank arms. The COM of the wheel carriers moves on an orbit around the y axis.
Semi Trailing Arm	Uses the same principle as the crank axle. The difference is that the rotating axis is not along the y-axis. Through this different orientation the wheel practises a camber change.
Linear 1 DOF	Linear uni-dimensional kinematics. The kinematics only depend on the wheel travel.
Linear 2 DOF	Linear two-dimensional kinematics. The kinematics depend on the wheel travel and the movement of the opposite wheel. This mode can be used to parameterize a rear axle.
External File	Non-linear kinematics. The kinematics are parameterized by the kinematics tables written to an external file.  In this case the values of each variable are defined for numerous steps of the wheel travel (e.g. from max. to min. bouncing, at step size 5mm).

- The various *Linear X DOF* models are defined by two values (see [Table 5.99: Variables for linear kinematic description](#)):
  - *Static*: the wheel center position when the trailer is standing on the road (e.g. some suspensions use static camber or toe-in)
  - *Compr.:* a coefficient defining the gradient of the wheel movement over wheel travel
  - *Oppos:* a coefficient defining the gradient of the wheel movement over wheel displacement of the opposite wheel
- The *Linear 2 DOF* models are defined by two values:
  - The kinematics will depend on the wheel travel and the movement of the opposite wheel (rigid or semi-rigid axle).
- The *Linear 1 DOF* model must only be used for a rear axle: the kinematics only depend on the wheel travel.
- With the non-linear model *External File*, you parameterize exactly the same variables as with the other models (tx, ty,... see [Table 5.99: Variables for linear kinematic description](#)). But in this case, no single ratio is defined between the variation of this parameter (e.g. camber angle rx) and the variation of the generalized coordinates (e.g. wheel travel), but you describe the absolute values of this variable for numerous steps of the generalized coordinates (e.g. wheel travel).

### Kinematics Models Linear and External File: Parameter Description

**tx, ty, tz (m)  
rx, ry, rz (rad)  
ISpring, IDamp,  
IBuff, LStabi (m)** Regardless of the mode you choose, the following 10 variables need to be parameterized as they are used by the model to define the kinematics. These parameters have a static value ( $c_{\text{static}}$  in [Figure 5.125](#)) and a coefficient describing the change according to the wheel travel and possibly according to the opposite wheel travel.



In case of a linear kinematics description this coefficient is the gradient of the linear correlation (see also [Table 5.100: Coefficients for the linear kinematics model](#)). In case of a non-linear kinematics description using *External File*, the absolute values are given. The following table gives an overview of the required parameters

Table 5.99: Variables for linear kinematic description

Parameter	Description	Unit (Static)	Unit (Compress)	Unit (Opposite)
Translation tx	wheel base variation	m	$m/m_{\text{wheel travel}}$	$m/m_{\text{opposite wheel travel}}$
Translation ty	track variation	m	$m/m_{\text{wheel travel}}$	$m/m_{\text{opposite wheel travel}}$
Translation tz	wheel travel variation	m	$m/m_{\text{wheel travel}}$	$m/m_{\text{opposite wheel travel}}$
Rotation rx	camber variation	rad	$\text{rad}/m_{\text{wheel travel}}$	$\text{rad}/m_{\text{opposite wheel travel}}$
Rotation ry	spin angle variation	rad	$\text{rad}/m_{\text{wheel travel}}$	$\text{rad}/m_{\text{opposite wheel travel}}$
Rotation rz	toe variation	rad	$\text{rad}/m_{\text{wheel travel}}$	$\text{rad}/m_{\text{opposite wheel travel}}$
Deflection ISpring	spring length variation	m	$m/m_{\text{wheel travel}}$	$m/m_{\text{opposite wheel travel}}$
Deflection IDamp	damper length variation	m	$m/m_{\text{wheel travel}}$	$m/m_{\text{opposite wheel travel}}$
Deflection IBuff	buffer length variation if activated	m	$m/m_{\text{wheel travel}}$	$m/m_{\text{opposite wheel travel}}$
Deflection IStabi	Stabilizer length (showing the vertical movement of the ARB at the point where the stiffness was measured) or angle variation, according to which unit you used for the ARB stiffness.	m or rad	$m/m_{\text{wheel travel}}$ or $\text{rad}/m_{\text{wheel travel}}$	$m/m_{\text{opposite wheel travel}}$ or $\text{rad}/m_{\text{opposite wheel travel}}$

**Linear X DOF** Let us now look in detail at the definition of the kinematic coefficients which have to be defined in the table for the linear kinematics modes:

Table 5.100: Coefficients for the linear kinematics model

Column	Description
Static	The values in this column indicate the static design parameters, e.g. when the wheel travel is zero.  The values for tx, ty, and tz should remain zero here, as otherwise this would lead to a change in the position of the wheels and wheel carriers. Sometimes, a value different from zero can be defined, e.g. to check a setup variant quickly. For rx, ry, and rz the values written here define the static camber, caster and toe in/out.
Compress	The values in this column give the variation of a parameter (e.g. the camber angle rx, in rad), per meter of wheel travel variation.

Table 5.100: Coefficients for the linear kinematics model

Column	Description
Opposite	The values in this column give the variation of a parameter (e.g. the camber angle rx in rad), per meter of wheel travel of the opposite wheel. This column is used only for a rigid axle (mode <i>Linear 2 DOF</i> for a rear axle).



Besides the translation of the wheel center, the change in length of spring, damper, buffer and ARB has to be parameterized. As you have seen, you have to parameterize the behavior of the ARB. As explained in section '[Anti Roll Bar \(ARB\)](#)' on page 312, you can define the stiffness of the ARB in two different units:

- Stiffness in N/m: in this case you have to parameterize IStabi in m/m<sub>wheeltravel</sub>
- Stiffness in Nm/rad: in this case you have to parameterize IStabi in rad/m<sub>wheeltravel</sub>



Once you have parameterized the kinematics, we suggest you to check the corresponding kinematics diagrams in the Model Check (see the [section 6.4 'Model Check'](#) on page 360).

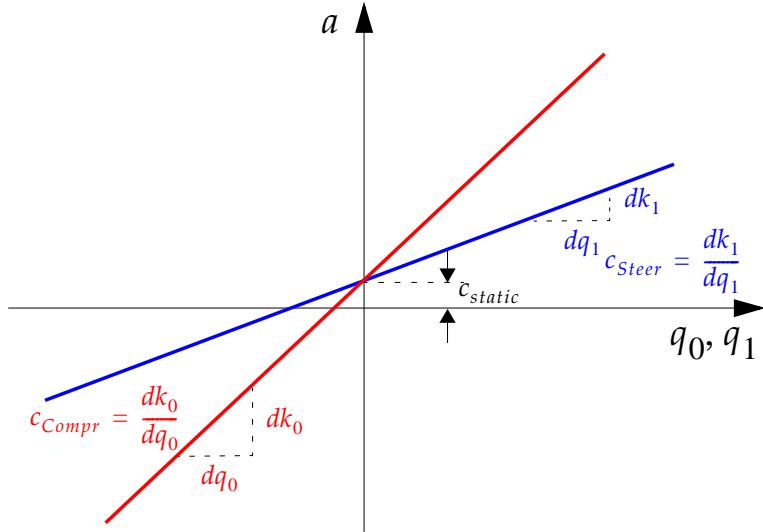


Figure 5.125: Definition of kinematics coefficients for the linear kinematics models

**External File**

The model *External File* basically requires the same input as the *Linear* models: static values and their variation according to the wheel travel (or the opposite wheel travel for a rigid or semi-rigid axle) of all of the 10 kinematics variables.

## Suspension Compliance

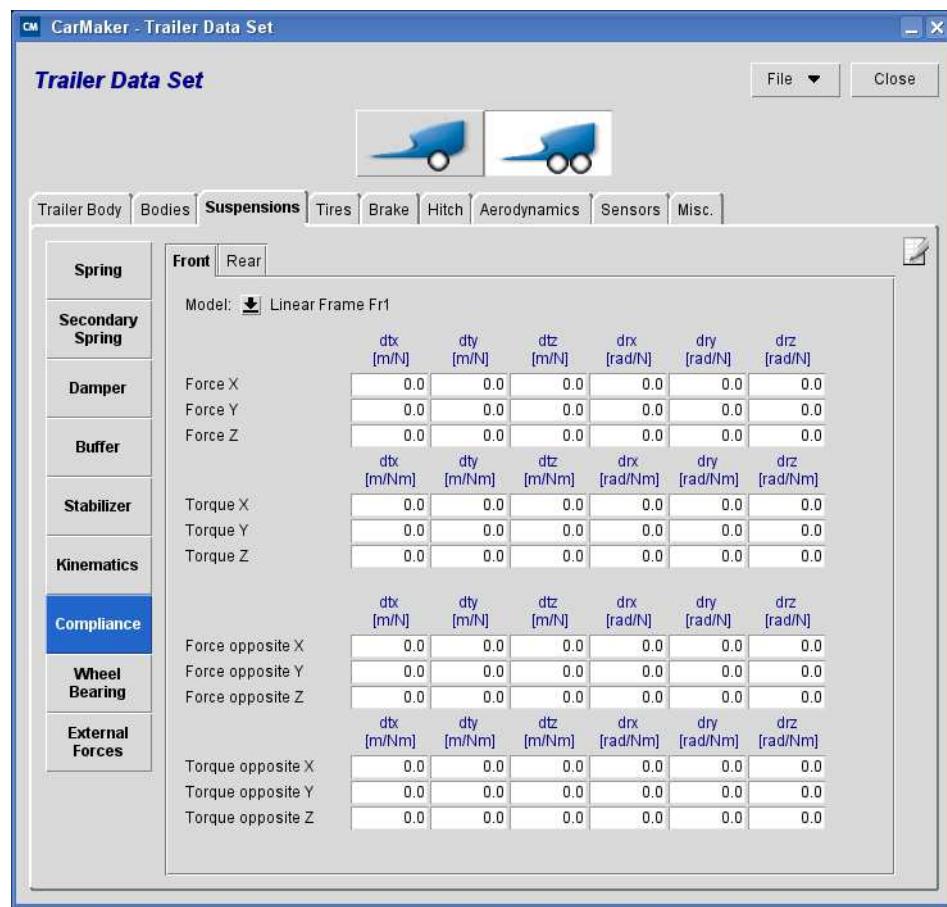


Figure 5.126: Definition of the linear compliance model in CarMaker

The compliance model takes into account the transition of the wheel when forces are applied due to elasticity effects of the suspension.

Defining the compliance in CarMaker is optional, which means you can simulate only with the kinematics and without compliance. Thus, you can define the compliance for 0, 1 or 2 axles.

As for the kinematics, you can choose between various compliance descriptions.

**Model** You can choose between the following compliance descriptions. Some of them are available in the *Model* option of the compliance tab, for others you have to select the mode not specified and define them in an external kinematics file:

Table 5.101: Compliance Models

Mode	Description	Available under Mode
not specified	No compliance is activated, except if it is already defined in the file describing the non-linear kinematics.	yes
Left	The compliance displacements caused by forces/torques acting on the wheel carrier are defined by the force/torque-compression rate.	yes

Table 5.101: Compliance Models

Mode	Description	Available under Mode
Left and Right	The compliance displacements caused by forces/torques acting on the current and the opposite wheel carrier are defined by the force/torque-compression rate.	yes
Linear Frame Fr1 or Linear Frame Fr2	The compliance displacements caused by forces/torques acting on both wheels are specified by linear coefficients. The forces are given either in <i>Fr1</i> or in <i>Fr2</i> .	yes
Coeff1DFr1 or Coeff1DFr2	The compliance displacements caused by forces/torques acting on the wheel carrier only are defined by linear coefficients at different wheel compressions. The forces are given either in <i>Fr1</i> or in <i>Fr2</i> .	no, file only
Displace1DFr1 or Displace1DFr2	The compliance displacement caused by forces/torques acting on the current or the opposite wheel carrier are defined by absolute values depending on the forces/torques applied.	no, file only
Displace2DFr1 or Displace2DFr2	The compliance displacement caused by forces/torques acting on the current or the opposite wheel carrier are defined by absolute values depending on the wheel compression and the forces/torques applied.	no, file only



Note that a non-symmetric parameterization is only possible with the compliance models *Coeff1DFr1/2*, *Displace1DFr1/2* and *Displace2DFr1/2*.

#### Definition of Compliance Coefficients for the models Linear Frame Fr1/2 and Coeff1DFr1/2

**Frc.x/y/z  
FrcOpp.x/y/z  
(N/m) or (N/rad)**

The compliance models *Linear Frame Fr1* or *Linear Frame Fr2* can be parameterized directly in the CarMaker GUI.

For the models *Coeff1DFr1* and *Coeff1DFr2* there is no GUI available. The model parameters are added to the kinematics description file (see [section 'External File' on page 316](#) and in the Reference Manual section 11.4.2 '*Coeff1DFr1*' and '*Coeff1DFr2*').

However, for both model the definition of the compliance coefficients is the same:

The variables Force\* and Torque\* correspond to the effect of a stress in the associated direction. For instance:

- *Force X* in column *dtx*: effect of a force applied to the wheel center in driving direction (x direction).
- *Torque X* in column *drx*: effect of a torque applied along the x-axis on the camber angle (*rx*).
- *Torque Y* in column *dtx*: effect of a torque applied along the y-axis on the wheel position in the x direction.

The variables Force\_opp\* and Torque\_opp\* represent the effect of a stress on the opposite wheel in the associated direction. For instance:

- *Force opp. X* in column *tx*: effect of a force applied forward on the opposite wheel, on the wheel position in the x direction.

- *Force opp. X in column ty*: effect of a force applied forward on the opposite wheel, on the wheel position in the y direction.

Table 5.102: Coefficients for the linear compliance model

Parameter	Description	Unit
dtx	wheel base variation due to force / torque applied	$m_{\text{Wheel travel}}/\text{N}_{\text{Force at wheel center}}$ or $m_{\text{Wheel travel}}/\text{Nm}_{\text{Torque at wheel center}}$
dty	track variation due to force / torque applied	$m_{\text{Wheel travel}}/\text{N}_{\text{Force at wheel center}}$ or $m_{\text{Wheel travel}}/\text{Nm}_{\text{Torque at wheel center}}$
dtz	wheel travel variation due to force / torque applied	$m_{\text{Wheel travel}}/\text{N}_{\text{Force at wheel center}}$ or $m_{\text{Wheel travel}}/\text{Nm}_{\text{Torque at wheel center}}$
drx	camber variation due to force / torque applied	$\text{rad}_{\text{Wheel rotation}}/\text{N}_{\text{Force at wheel center}}$ or $\text{rad}_{\text{Wheel rotation}}/\text{Nm}_{\text{Torque at wheel center}}$
dry	spin angle variation due to force / torque applied	$\text{rad}_{\text{Wheel rotation}}/\text{N}_{\text{Force at wheel center}}$ or $\text{rad}_{\text{Wheel rotation}}/\text{Nm}_{\text{Torque at wheel center}}$
drz	toe variation due to force / torque applied	$\text{rad}_{\text{Wheel rotation}}/\text{N}_{\text{Force at wheel center}}$ or $\text{rad}_{\text{Wheel rotation}}/\text{Nm}_{\text{Torque at wheel center}}$

### Coeff1DFr1/2, Displace1DFr1/2 and Displace2DFr1/2 compliance models

For both models, the compliance description has to be parameterized in the file chosen for the *MapNL* kinematics mode ([section 'External File' on page 316](#)).

In order to know how you can use those modes, please read the Reference Manual section '[Coeff1DFr1](#)' and '[Coeff1DFr2](#)' and section '[Displace1DFr1](#)' and '[Displace1DFr2](#)'.

## Wheel Bearing Friction

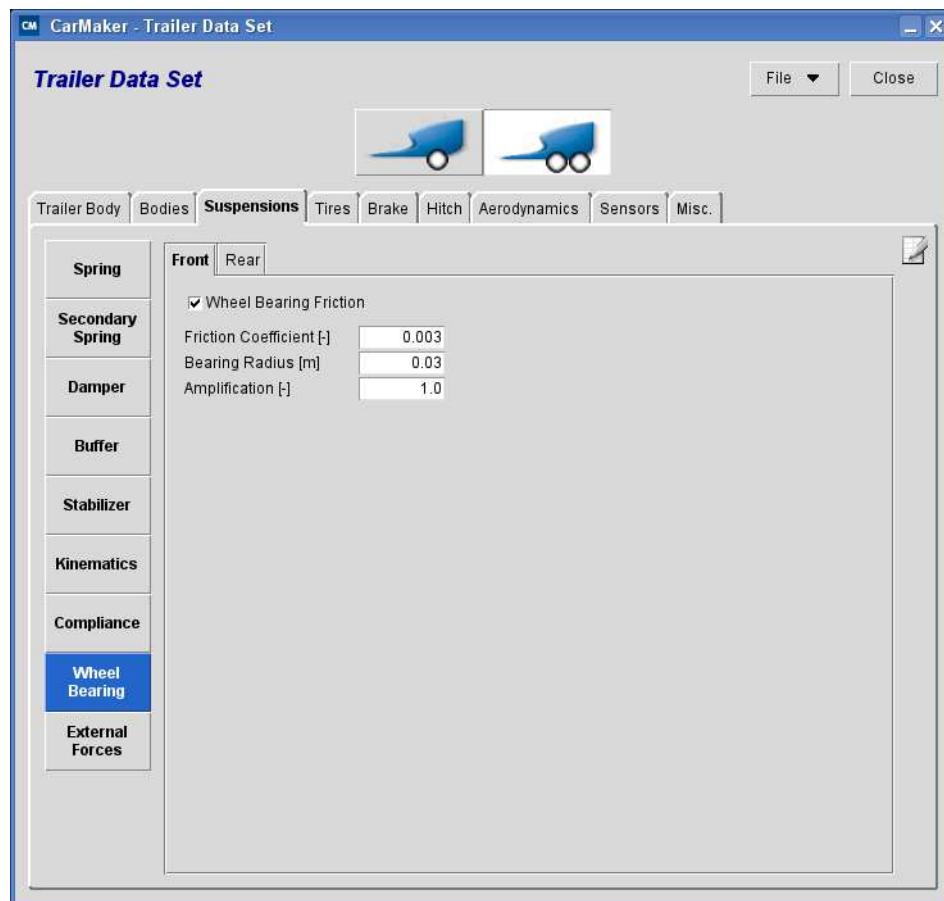


Figure 5.127: Defining a friction torque at the wheel bearings

For a given axle, you have the possibility to take the friction torque at the wheel bearings into account. The friction torque is calculated as follows:

$$T_{Friction} = \mu \cdot F_{Friction} \cdot R = \mu \cdot F_{Axe} \cdot \mu \cdot R$$

- |                                 |   |
|---------------------------------|---|
| <b>Wheel Bearing Friction</b>   | This check box activates/deactivates the effect of friction at the wheel bearings. If activated, other parameters appear.   |
| <b>Friction Coefficient (-)</b> | The wheel bearing friction is characterized by a single coefficient $\mu$ , which defines the friction force on the bearing. This value should be delivered by the bearing manufacturer.                                    |
| <b>Bearing Radius (m)</b>       | The bearing radius $R$ enables to evaluate the friction torque on the bearing based on the friction force. This parameter can be found out by asking the bearing manufacturer or by measuring it directly on the component. |
| <b>Amplification (-)</b>        | Using the amplification factor $\mu$ the friction coefficient can be scaled very quickly instead of modifying the parameter itself.   |

## Parametrizing the External Forces

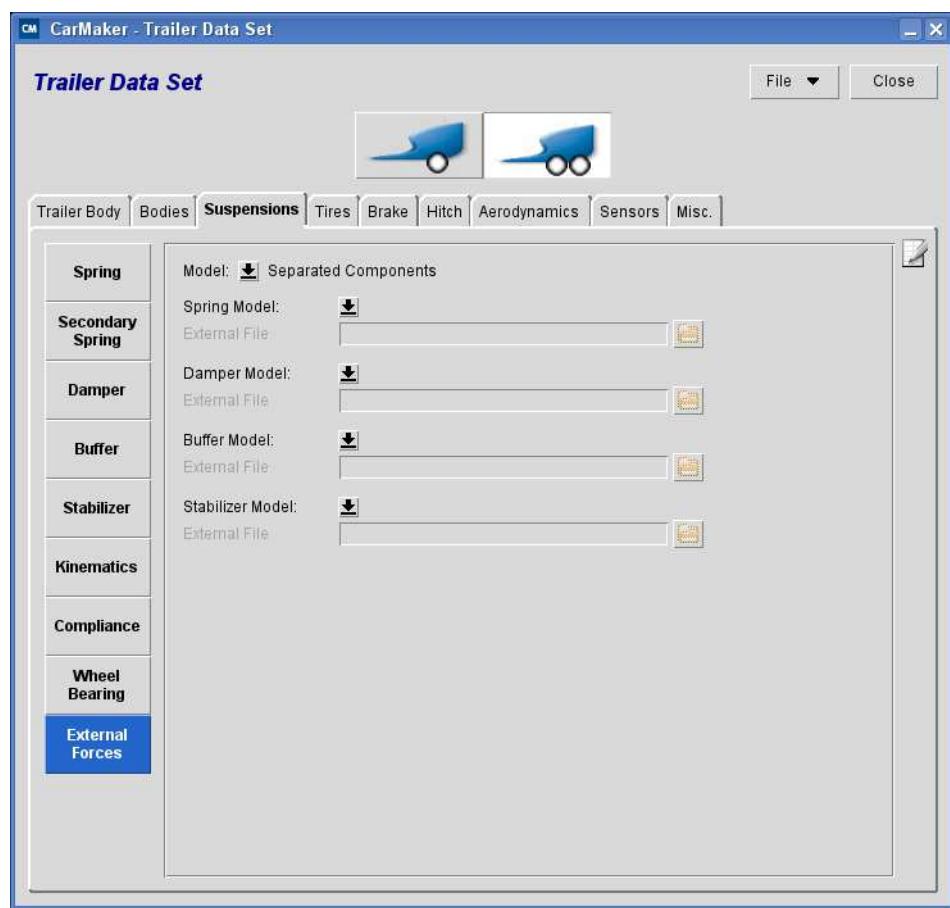


Figure 5.128: Interface for external suspension forces

This is an interface for very specific applications. Forces calculated by an external model can be added to the spring, damper or stabilizer forces calculated by CarMaker. The model can be generated by a c-code or using Matlab Simulink. For the latter, it can be necessary to read some user specific parameters from file to configure this model. This parameter file can be selected this dialog.

Please find further information in [section 5.14 'Suspension: External Forces'](#) on page 201.

## 5.26.5 Brake

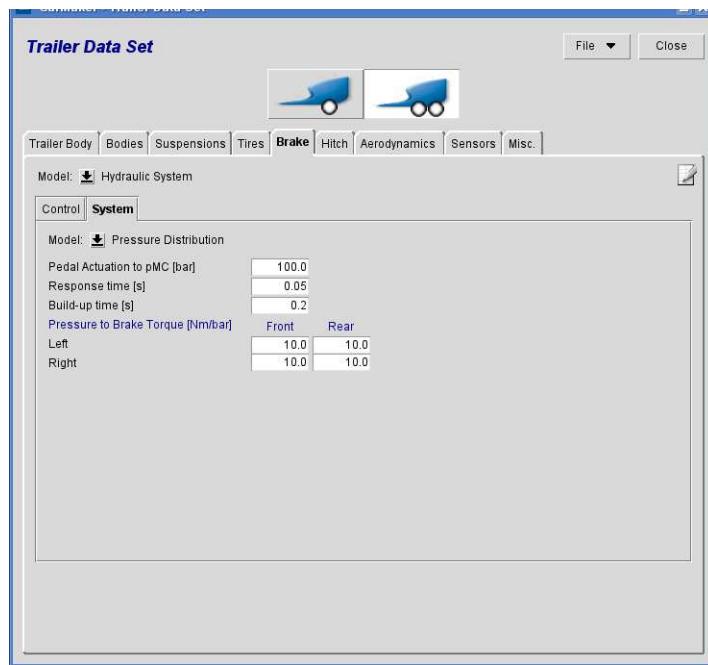


Figure 5.129: Parameters of the brake system

The brake model determines the ratio between the force at the brake pedal to the braking torque at each wheel.

**Model** Here it is defined, which brake model is used:

Table 5.103: Brake Models

Model Kind	Description
Overrun Brake	Brake torque proportional to drawbar force.
Overrun Brake with Friction	Brake torque based on friction.
Hydraulic System	Hydraulic brake system model including brake control unit and hydraulic system, several configurations available

### Model: Overrun Brake

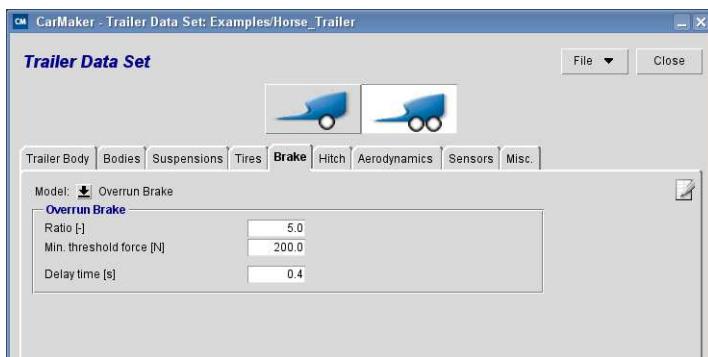


Figure 5.130: Parameters of the Overrun Brake

- Ratio (-)** This ratio defines the relation between the total brake forces at all wheels to the force of the trailer hitch.
- Min. threshold force (N)** This threshold corresponds to the minimum force at trailer hitch for which the overrun brake becomes active.
- Delay time (s)** This delay defines the time needed to build up brake force at the wheels.

### Model: Overrun Brake with friction

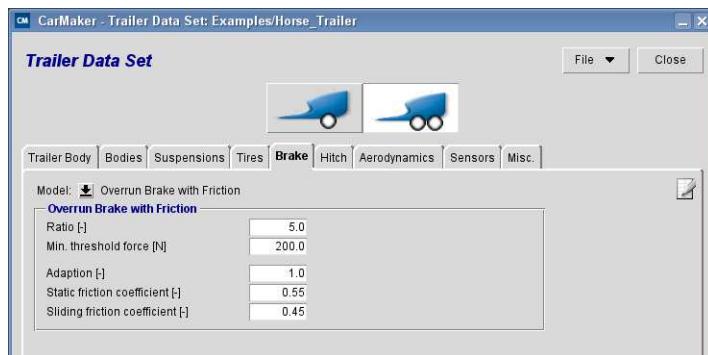


Figure 5.131: Parameters of Overrun Brake with Friction

- Ratio (-)** This ratio defines the relation between the total brake forces at all wheels to the force of the trailer hitch.
- Min. threshold force (N)** This threshold corresponds to the minimum force at trailer hitch for which the overrun brake becomes active.
- Adaption (-)** This factor can be used to alter the trailer mass for stick-slip calculation/
- Static friction coefficient (-)** This friction coefficient applies for static friction between brake pads and brake disk.
- Sliding friction coefficient (-)** This friction coefficient applies for sliding friction between brake pads and brake disk

### Model: Hydraulic System



Figure 5.132: Parameters of the Hydraulic Brake model for trailer

The *Hydraulic System* brake model for trailer is similar to the *Hydraulic System* model for the vehicle. Please read [section 5.16 'Brake'](#) for more information about this model. Of course, the use of Regenerative Brake Modes *Parallel* and *Serial* only makes sense if there is a generator that can recuperate energy at the trailer wheels (e.g. an additional electric motor).

## 5.26.6 Hitch

The trailer model is connected to the towing vehicle by a virtual spring-damper-system. The point of contact is in the middle of the coupling device. The parameters are determined automatically depending on the masses of the vehicles. The parameters are chosen in a way that even with huge coupling forces the relative travel of the coupling contact points is about a few millimeters and numerical stability is ensured.

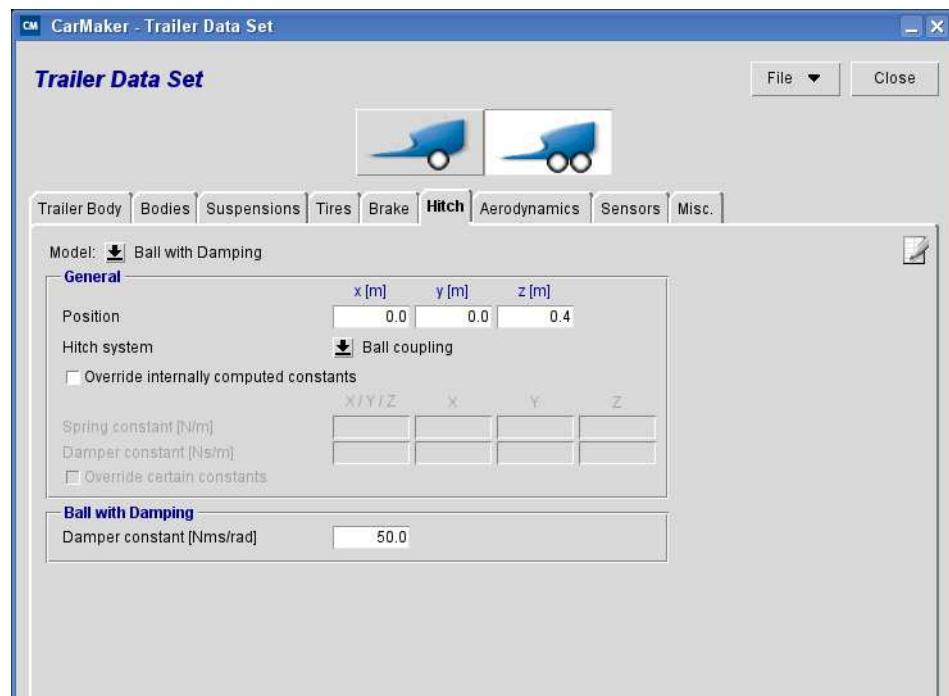


Figure 5.133: Parameters of the hitch system

To ensure stabilization of the trailer, different hitch improvements have been developed. The CarMaker trailer model supports the following hitch models:

Table 5.104: Hitch Models

Mode	Description
Ball	No stabilization, normal ball joint
Ball with damping	Ball joint with articulation angle dependent hydraulic damping
Ball with friction	Ball joint hitch with friction damper
Ball with trapezoid	Four joint hitch
Truck generic	Fifth wheel coupling used with trucks

**Hitch Position** Center position of the trailers hitch. Coordinates are specified in trailers FrD axis system.

- Override internally computed constants** Optional. Overwrites the internally computed spring/damper constant for trailer hitch. With this parameter no real values for the trailers hitch spring constant should be used rather than values that ensure numerical stability. Default: a default is computed internally by the masses of towing vehicle and trailer. Can be defined in all direction of space or in one certain direction.

## 5.26.7 Aerodynamics

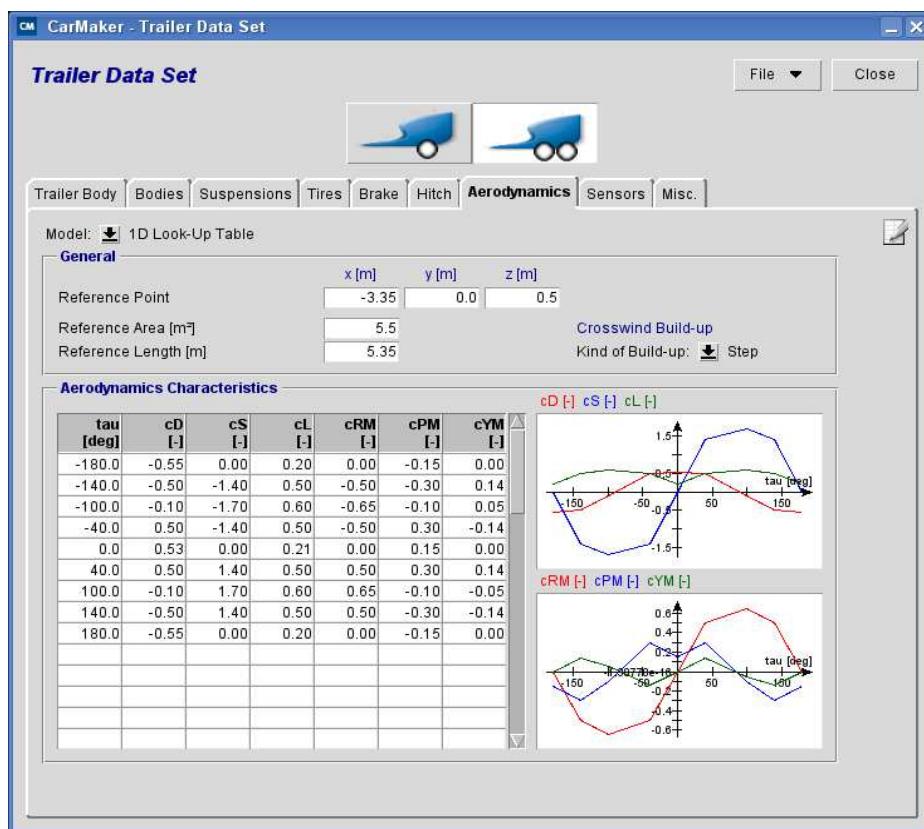


Figure 5.134: Definition of the aerodynamics parameters

The aerodynamic model applies additional variable forces to the trailer body according to the trailer speed. The adaptation of the map by accurate values is often not required as the aerodynamic model is of minor importance at low speeds. Please note, that slipstream effects of preceding vehicles are not considered.

If you need a better aerodynamic model that takes into account the ride height, please contact the IPG Service-Team under [CarMaker-Service@ipg.de](mailto:CarMaker-Service@ipg.de) to receive information about a more detailed aerodynamic model. However, this model already enables to reach very good results, and that is why it has been used successfully even by our customers working on racing applications.

- Model** By default, only one aerodynamic model is available in CarMaker. The single default option available is the model *Look-Up Table 1D*. Additional information about the aerodynamical model, especially about the equations used to calculate the aerodynamic forces, can be found in the Reference Manual, section 'Aerodynamics'.

In case you would like to use your own aerodynamic model (built in Simulink or C-Code), or you would like to exclude the aerodynamic influences on your trailer completely, then the option *not specified* should be chosen for *Model*.

The following general parameters need to be defined for the aerodynamic model *Look-Up Table 1D* (all coordinates refer to FrameD).

**Aero Marker (m)** Caution: it is the single aerodynamical parameter that is to be parameterized in the tab *Bodies!* This point is used by the module that simulates side winds: see [section 'Wind' on page 62](#).

When the trailer is gripped by side wind, the wind starts to take effect only from a certain point on, e.g. if only the bumper is attacked by side wind the driver will usually not recognize the effects. Ahead of this point, the trailer body does not offer enough contact surface to the wind to take effect.

It is that very point that is parameterized by the *Aero Marker*. Please note that it is not the point where the aerodynamic forces apply! It is only the point from which the forces created by the side winds are taken into account. But the forces themselves are applied to another point (see Reference Point [m]).

We suggest to choose this point according to the shape of the trailer body.

**Reference Point (m)** It is the center of pressure, the point where the aerodynamic forces, including the side wind forces, are applied.

If you do not have any information on the exact location of this point the trailer's center of gravity can be taken instead. According to the shape of the trailer body, you can move this point so that the aerodynamic influence in the simulation approaches to what you observe on the track.

**Reference Area (m<sup>2</sup>)** The trailer *Reference Area* is the projected frontal area including tires and underbody parts.

**Reference Length (m)** Per default the *Reference Length* is the same as the wheel base of the trailer. It is used to calculate the torques applied to the Reference Point due to aerodynamic effects. If you wish to tune this value, have a look at (EQ 76) in the Reference Manual.

**Aerodynamics Coefficients (-)** The *Aerodynamic Coefficients* used by the model *Look-Up Table 1D* have the following meaning:

Table 5.105: Aerodynamic Coefficients

Parameter	Description	Unit
tau	Wind angle	deg
cD	Coefficient used to calculate the aerodynamical drag force (x-axis).	-
cS	Coefficient used to calculate the aerodynamical side force (y-axis).	-
cL	Coefficient used to calculate the aerodynamical lift force (z-axis).	-
cRM	Coefficient used to calculate the aerodynamical roll moment (around x-axis).	-
cPM	Coefficient used to calculate the aerodynamical pitch moment (around y-axis).	-
cYM	Coefficient used to calculate the aerodynamical yaw moment (around z-axis).	-

Please find more information about the formulas that use above coefficients in the Reference Manual.

The values used for a zero wind angle are probably the most important ones since they will always be used during the simulation: driving the trailer causes a virtual front wind. Other angle values are used especially if side wind occurs ([section 'Wind' on page 62](#)). In particular, the value of the parameter  $cD$  for  $\tau = 0$  deg is the well known  $c_x$  value.

## 5.26.8 Sensors

In the *Sensors* tab, there is the possibility to define three types of sensors: *Side Slip Angle Sensors*, *Body Sensors*, and *Driver Assistance Sensors*.

### Side Slip Angle Sensors

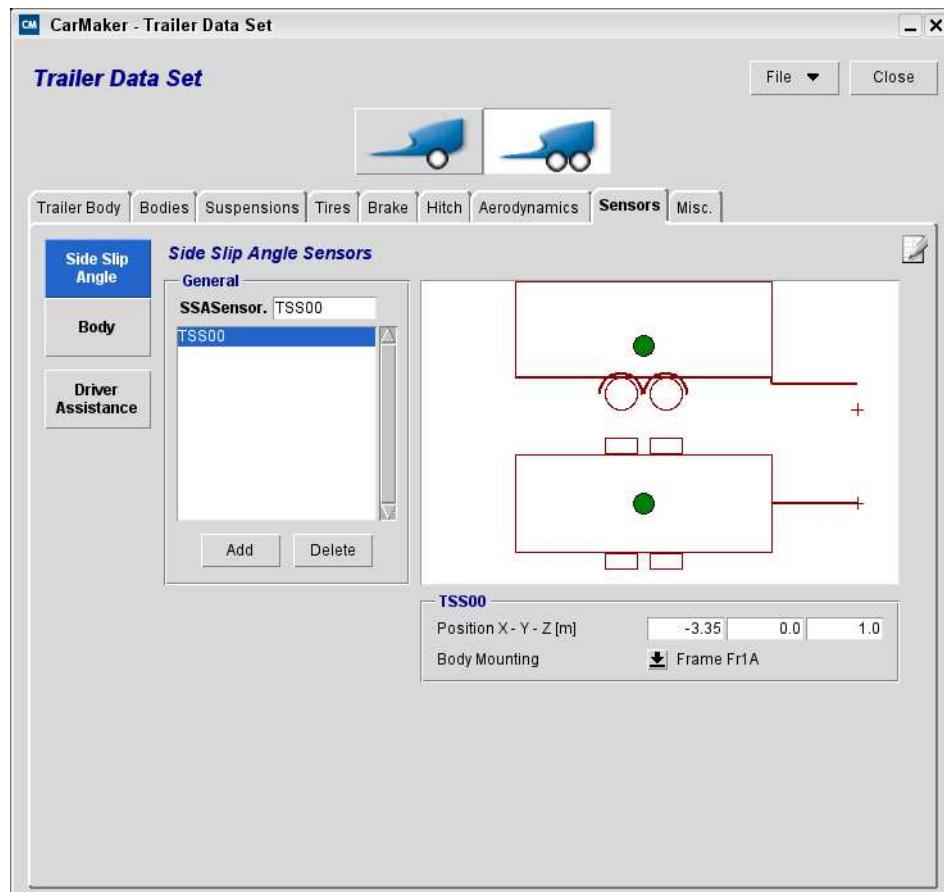


Figure 5.135: Definition of *Side Slip Angle Sensors*

Here you have the possibility to define numerous side slip angle sensors. At least one must be specified, which is located in the trailer's center of gravity per default.

The parameters of the *Side Slip Angle Sensor* are as follows:

- SSASensor.** Here you can enter a name for the current sensor, to alleviate the identification of the sensor quantities, e.g. in IPGControl. All User Accessible Quantities of this sensor will start with the prefix `Car.SSASensor.Name.*`. In the example in [Figure 5.135](#), the UAQs of this sensor will start with `Car.SSASensor.A00.*`.
- Position X-Y-Z (m)** The location of the *Side Slip Angle Sensor* is defined by coordinates in *FrameD*. The sensor position can be visualized on the graphical representation of the trailer.

**Body Mounting**

The mounting point of the *Side Slip Angle Sensor* is defined here. It is always the trailer based coordinate system *Fr1A*, in case a *Flexible Body* is used you can choose between *Fr1A* and *Fr1B* (see [section 5.26.1 'CarMaker Coordinate Systems' on page 303](#) for information about the axis systems and section [section 5.26.2 'Bodies' on page 304](#) for more information on flexible bodies).

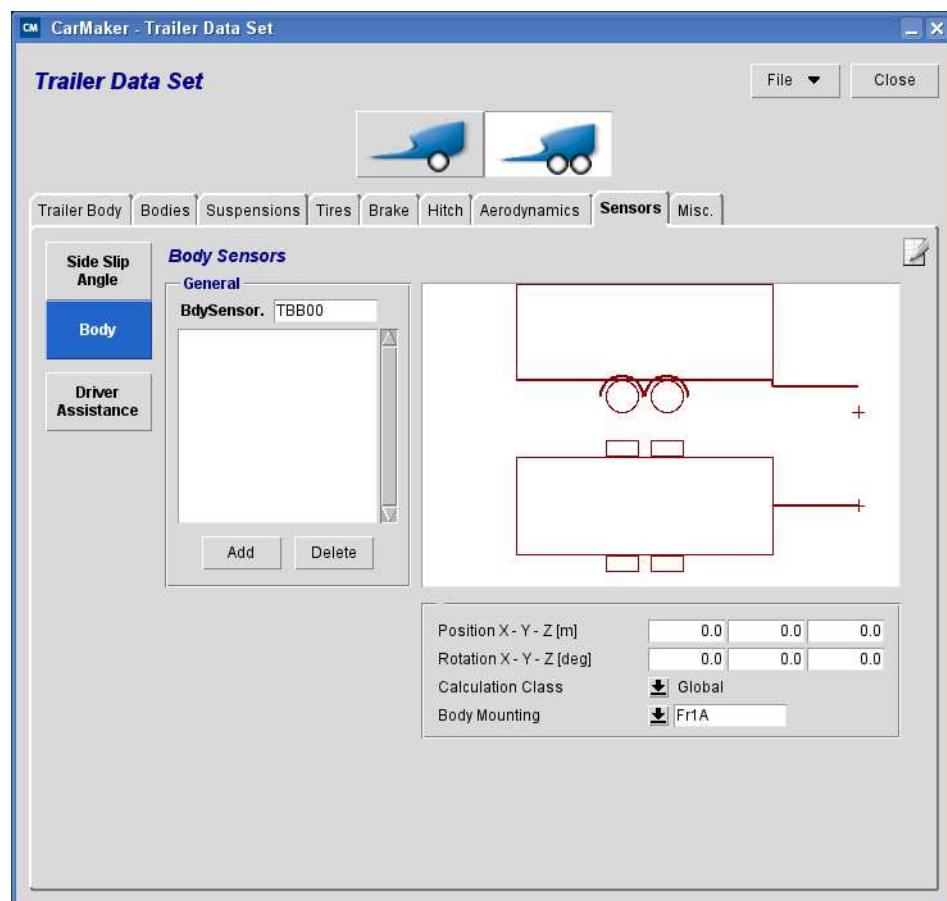
**Body Sensors**

Figure 5.136: Definition of *Body Sensors*

A *Body Sensor* is an inertial sensor that can be positioned anywhere on the trailer to measure the position, velocity, rotational velocity, acceleration and rotational acceleration. Up to 10 body sensors can be placed on the trailer. A small set of parameters enables to parameterize each of them:

**BdySensor.** Here you can enter a name for the current sensor, to alleviate the identification of the sensor quantities, e.g. in IPGControl. All User Accessible Quantities of this sensor will start with the prefix *Car.BdySensor.Name.\**. In the example in [Figure 5.136](#) the UAQs of this sensor will start with *Car.BdySensor.B00.\**.

**Position X-Y-Z (m)** These coordinates indicate the position of the selected sensor on the trailer. They are interpreted in the *FrameD*. If the body sensor is mounted on the wheel carrier, the parameters describe the position before the static equilibrium configuration. The sensor position can be visualized on the graphical representation of the trailer.

**Rotation X-Y-Z (deg)** These angles specify the rotation of the sensor frame according to the frame on which the sensor is mounted. Rotation order: ZYX.

**Calculation Class** This parameter specifies which frame should be used as reference for the calculation of displacements, velocity and acceleration in the sensor position. The following options are available:

Table 5.106: Description of the *Calculation Classes*

Calculation Class	Description
Global	All velocities and accelerations are calculated in global frame Fr0
Global + Body Mounted	All velocities and accelerations are calculated in the global frame Fr0 and in the local frame on which the body sensor is mounted.
Global + Body Mounted no G	All velocities and accelerations are calculated in the global frame Fr0 and in the local frame on which the body sensor is mounted. The accelerations in the local frame do not consider the gravitation of the earth.

**Mounting** The parameter indicates the frame on which the selected body sensor is mounted. A detailed description on the meaning of the frames can be found in [section 5.26.1 on page 303](#).

Table 5.107: Description of the *Mounting* positions

Mounting	Description
Frame Fr1A/B	Mounted on the main body; in case of a flexible body the sensor can be mounted either on the front or rear body part (see <a href="#">section 5.26.2 on page 304</a> ).
Frame Fr2<pos>	Mounted on the wheel carrier frames, where <pos> stands for the wheel position (FL=front left, FR=front right, RL=rear right, RR=rear right).

For further details about Body Sensors, see the Reference Manual, section 'Body Sensors'.

## Driver Assistance Sensors

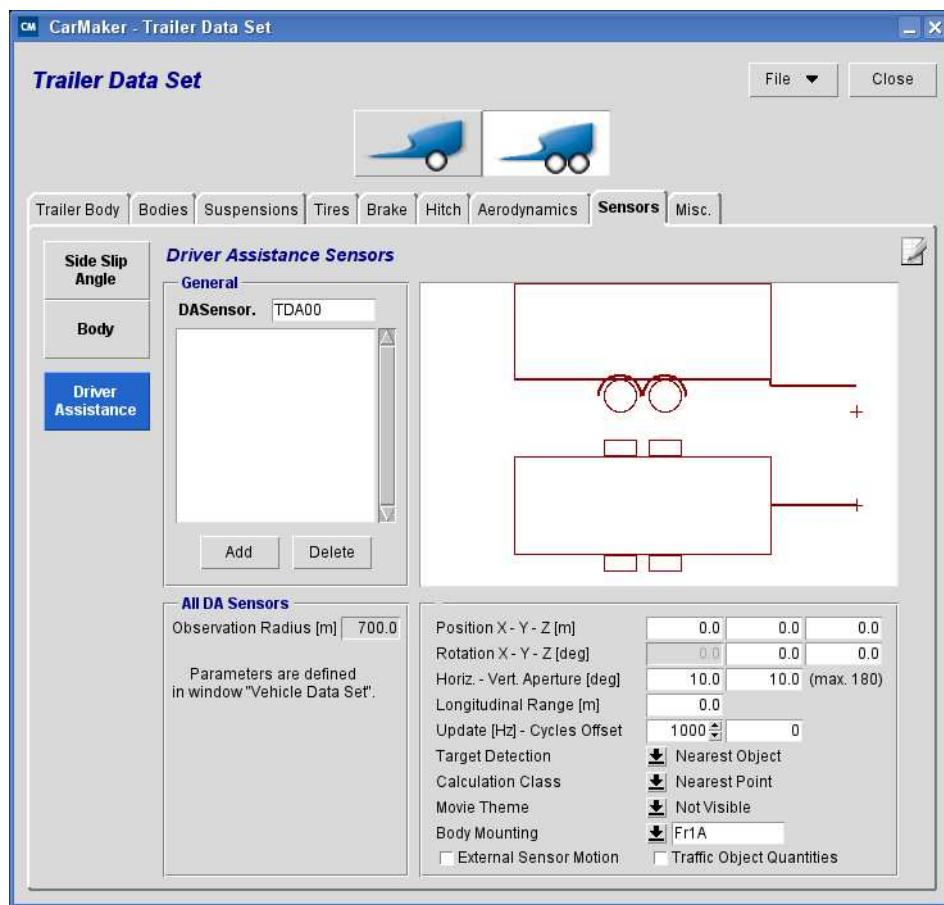


Figure 5.137: Definition of *Driver Assistance Sensors*

CarMaker offers the possibility to integrate up to 20 *Driver Assistance Sensors* in the trailer. Each one can be parameterized independently, just by selecting the sensor in the *General* area. The parameters that are specific to the selected sensor are displayed under the graphical representation of the trailer.

**DASensor.** Here you can enter a name for the current sensor, to alleviate the identification of the sensor quantities, e.g. in IPGControl. All User Accessible Quantities of this sensor will start with the prefix *DASensor.Name.\**. In the example in [Figure 5.137](#) the UAQs of this sensor will start with *DASensor.C00.\**.

**Observation Radius (m)** It specifies the radius (in meters) of the observation area around the sensor. No calculation will be done for the objects that are outside this area. Default: 500m.

**Position X-Y-Z (m)** These coordinates indicate the position of the selected sensor on the trailer. They are interpreted in the *FrameD*. The sensor position can be visualized on the graphical representation of the trailer.

**Rotation X-Y-Z (deg)** These angles specify the rotation of the sensor frame according to the trailer frame (*FrameD*). This specifies the viewing direction of the sensor. Rotation order: ZYX.

**Horizontal Aperture (deg)** This parameter defines the horizontal aperture of the sensor beam [deg], maximum angle is 180°deg.

<b>Vertical aperture (deg)</b>	This parameter defines the vertical aperture of the sensor beam [deg], maximum angle is 180°deg.
<b>Longitudinal Range (m)</b>	The longitudinal range of the sensor beam is defined here.
<b>Update (Hz)</b>	This parameter defines the frequency, with which the environment is scanned by the driver assistance sensor.
<b>Cycles Offset</b>	This parameter serves for performance optimization of the simulation. If you are using more than one sensor and not every ms is required for simulation, you can define an offset here.
<b>Target Detection</b>	This parameter specifies the sensor type:

Table 5.108: *Target Detection* modes for the *Driver Assistance Sensors*

Mode	Description
Lane Tracking	The detection of a relevant target is done by investigating the current trailer trajectory, the width of its driving lane and the distance to the object.
Nearest Object	The sensor module finds the closest object and calculates the distance from this object to the sensor. The object with the smallest overall distance is the relevant target.
None	No target detection.



Please note, that hill tops or fog will have no effect on the range of the sensor.

<b>Movie Theme</b>	These features are used for visualization purposes in IPGMovie only. They do not have any effect on the detection of obstacles.
<b>Calculation Class</b>	This parameter specifies if the calculated distance to the object refers to the reference point or to the nearest point:

Table 5.109: Different *Calculation Classes* for the *Driver Assistance Sensors*

Mode	Description
Reference Point	Calculation of relative quantities in reference point.
Relative Angles	Calculation of relative quantities in reference point AND in nearest point; calculation of relative angle between the object and the ego trailer (comparison of x-axis <i>Frame1</i> ).
Nearest Point	Calculation of relative quantities in reference point AND in nearest point.
Image Area	Calculation of relative quantities in reference point AND in nearest point; calculation of incidence angles and projections on image area.

<b>Additional Traffic Object Quantities</b>	If checked, additional quantities for ALL traffic objects and not only the detected obstacles are available in the <i>Data Dictionary</i> , e.g. you can access those quantities via IPGControl.
<b>Consider External Sensor Motion</b>	If checked, external relative sensor travel and rotation are considered. To use the external sensor motion, the DVA-quantities <i>DASensor.&lt;Sensor-Name&gt;.rx_ext</i> , <i>DASensor.&lt;Sensor-Name&gt;.ry_ext</i> and <i>DASensor.&lt;Sensor-Name&gt;.rz_ext</i> have to be defined.  For more information about the external sensor motion, please have a look at section 19.7 <i>Parameterization of DASensors</i> in the Reference Manual.

For further details about *Driver Assistance Sensors*, see Reference Manual, section 'Driver Assistance Sensor'. Definition of a *Road Property Sensor*.

## 5.26.9 Miscellaneous

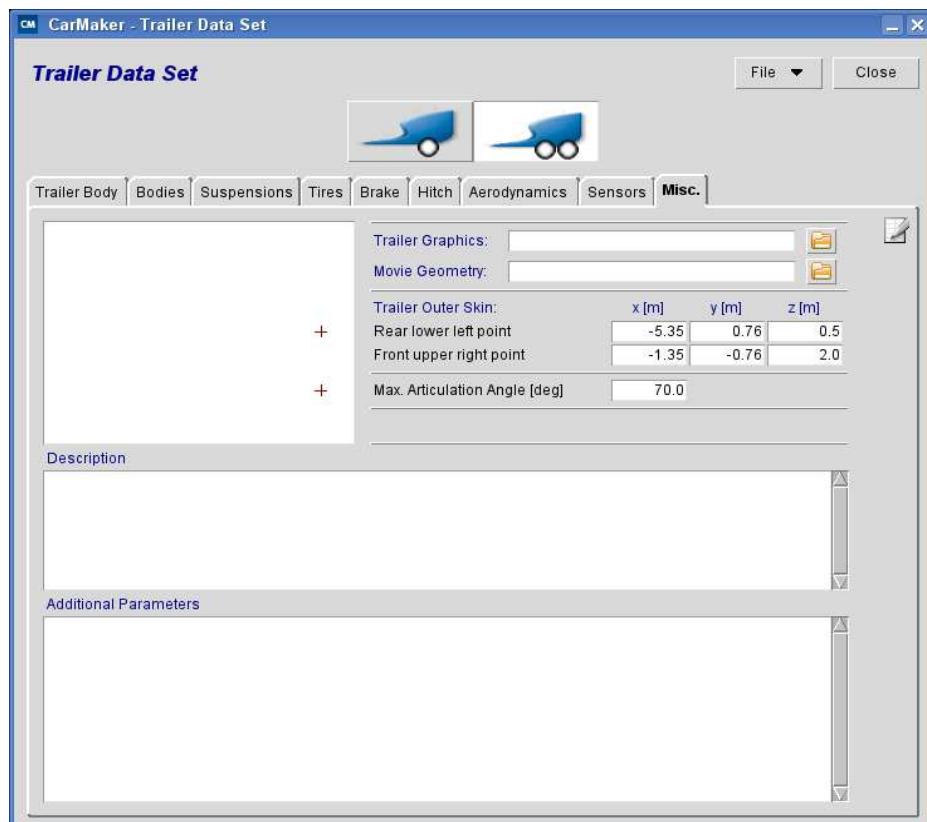


Figure 5.138: A few additional parameters can be defined.

- Trailer Graphics** By clicking on *Select* you can choose the picture that is displayed in the trailer editor. However, you can create your own trailer graphic, too. Supported file types are PNG images or a tcl/tk script.
- Movie Geometry** Here you choose the so called "movie object" that is used by IPGMovie for the animation. This movie object is used only for the trailer being simulated, not for the traffic (for this, read the [section 4.8.2 'Defining Traffic Objects' on page 128](#)). If you don't choose any movie object here, IPGMovie will use the ABRAHAS object by default. More information about the ABRAHAS object can be found in the [section 7.1.1 'Navigating with the camera' on page 398](#).
- Trailer Outer Skin (m)** Here the size of the ABRAHAS object displayed in IPGMovie can be defined. More information about the ABRAHAS object can be found in the [section 7.1.1 'Navigating with the camera' on page 398](#).
- Description** Here you can write some free comments. These comments are displayed at the bottom of the selection window when you load a trailer data set.
- Additional Parameters** If you have developed a special model, you may need to parameterize it. In this case, you can write some additional parameters in this field that will be written in the trailer file, and then read the additional parameters directly from the trailer file.

## 5.27 Tire Model

### 5.27.1 Selecting a Tire

The tires are parameters within the vehicle data set. On that reason, the tire data used has to be specified in the *Vehicle Data Set Editor* (*CarMaker Main GUI > Parameters > Vehicle*) where you can find a tab called *Tires*. The main advantage of this method is, that a suitable tire data set is already loaded with the test car selection.

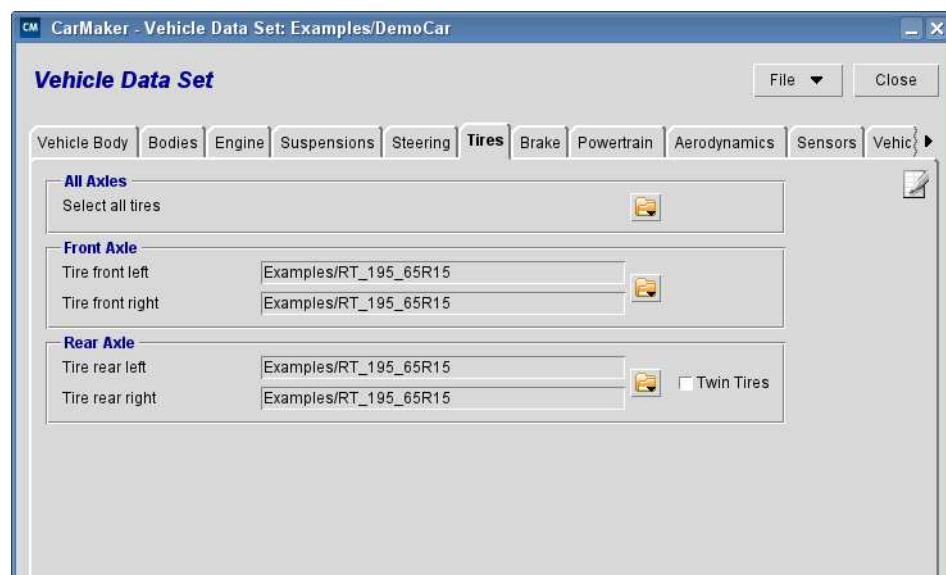


Figure 5.139: Selecting *Tires* in the Vehicle Dataset Editor

- All Axles** At this point, the tires for all axles of the vehicle can be set at once. To select a tire data set, click on the folder button at the right or into the entry field to open a file browser.
- Front Axle** Select the tire data for both wheels on the front axle. To select a tire data set, click on the folder button at the right or into the entry field to open a file browser.
- Rear Axle** Select the tire data for both wheels on the rear axle. To select a tire data set, click on the folder button at the right or in the entry field to open a file browser. Optionally, twin tires can be added to the rear axle by activating the tick box.
- Twin Tires Rear Axle** If you would like to simulate e.g. transporters which can carry high loads, you have the possibility to mount twin tires on the rear axle. Activating this option, the layout of the *Bodies* tab changes. Now the y-position of the wheel carriers of the rear axle is unlocked - with the position of the wheel carrier, you can determine the position of the twin tires. The wheel positions defines the outer wheel - the inner wheels are mirrored.

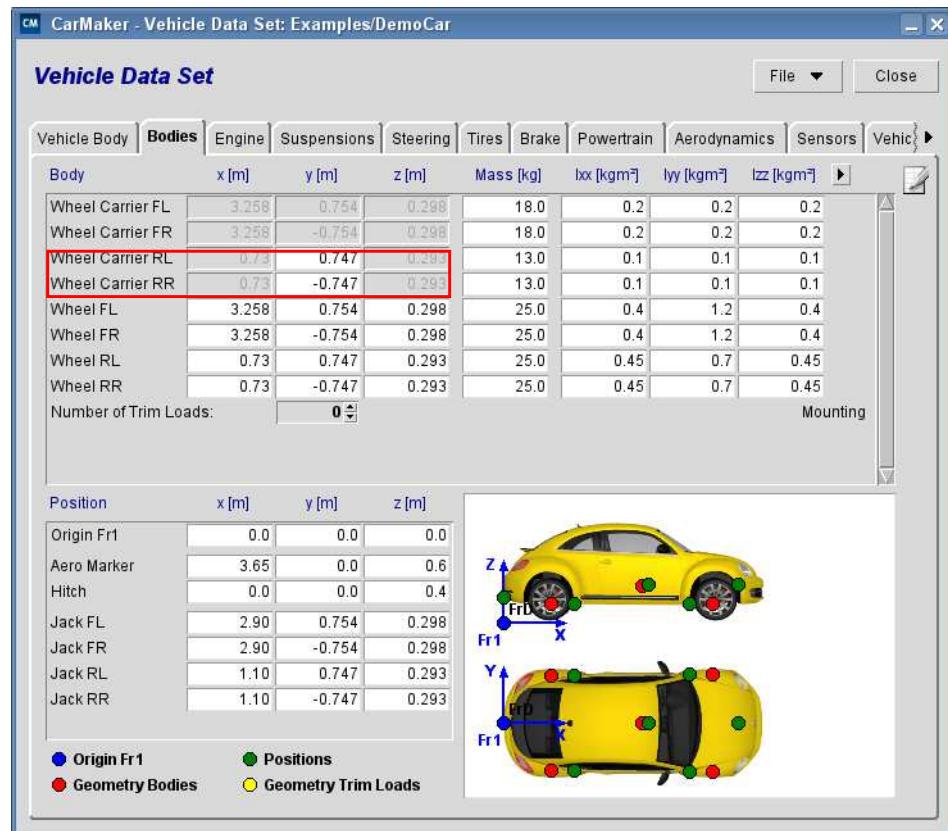


Figure 5.140: Positioning of the wheel carriers in case of twin tires



A second way to assign tires to your vehicle is in the CarMaker main GUI. This selection is stored in the TestRun configuration and overwrites the setting of the *Vehicle Data Set Editor*. It is used to overrule the default tire sets of the selected test car for the current simulation only. Opposed to a change of the tire selection in the vehicle data set, this setting does not effect all TestRuns using this test vehicle.

By clicking on *Select* in the main GUI, the data sets for all wheels can be assigned at once. By clicking on one tire name, you can modify the selection for this wheel independently on the others. A balloon help pops up when the cursor is moved over the item, to indicate the tire position:

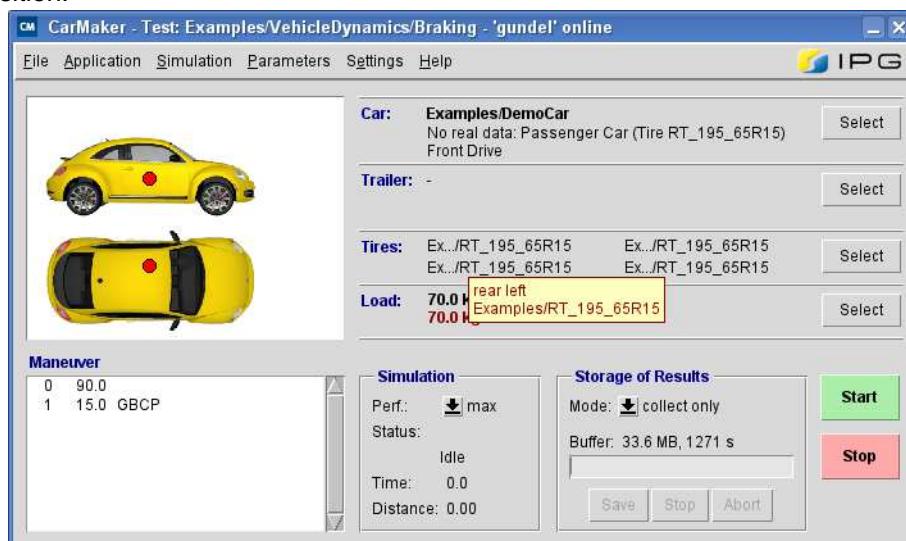


Figure 5.141: Balloon help indicating the tire position

Within the respective vehicle Infofile in the "Data/Vehicle" folder of your project directory, the tires are named as follows:

```
Front left = Tire.0
Front right = Tire.1
Rear left = Tire.2
Rear right = Tire.3
```

## 5.27.2 Overview of Tire Models

The purpose of the tire models is to calculate the slips and the stresses applied to the contact point on the road (patch point) and then to convert them into stresses applied to the wheel center, since the rest of the vehicle model directly uses those efforts.

By default, CarMaker supports four tire models:

Table 5.110: Supported Tire Models

Tire Model	Description
IPGTire	This model is based on the TYDEX-format (description of the digitized tire characteristics in a text file).
Pacejka 5.2 (Magic Formula)	Pacejka model, version 5.2 (description of the tire characteristics with mathematical parameters used in the equations of the model). The ADAMS property files are accepted, too.
MF-Tire/MF-Swift 6.1	To use the tire model MF-Tire/MF-Swift 6.1 by TNO, a corresponding dynamic library must be specified (currently only Windows version supported). The path for the library is set in the SimParameters file.
TameTire	Tame Tire is a physical tire model that has been developed by Michelin R&D. The model aims at an accurate description of the transient mechanical and thermal behavior of a tire.

The usage of each of the models will be described in the following subsections. Let us have a look at some advises that will help you to decide to use one or the other model.

- IPGTire is particularly recommended if you have the measured characteristics of the tire (e.g. lateral force vs. side slip).

With these characteristics, you are able to transfer them to a text file with respect to the TYDEX format. Then you just need to convert the text file into a binary format readable by CarMaker via a tool included in CarMaker.

The main drawback is that formatting the characteristics requires much time, and that the data can not be used directly (it has to be converted into a binary format).

- Pacejka is recommended if you wish to tune your tire data set fast and often.

In this case you just have to modify the Pacejka parameters in the text file where they are written, and start your simulation again. You do not need to convert any file to any other format.

The main drawback is the poor standstill capabilities of the model itself. IPG found a workaround to avoid disturbance. In addition, you can only use this model if you have generated the Pacejka parameters during the measurement procedure. You can not extrapolate them from a raw tire characteristics.

Additional information about the tire models, especially the axis system used by the tire calculations, can be found in the Reference Manual in the section *Tire*.

### 5.27.3 Tire Data Set Editor

There are two ways to launch the *Tire Data Set Editor*:

- Start the file browser to select a tire and click on *Edit* instead of *OK*:

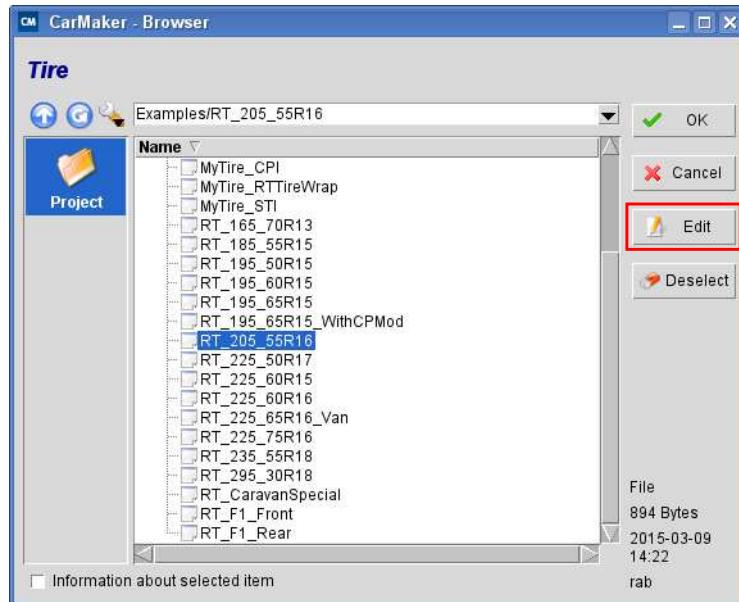


Figure 5.142: Starting the Tire Data Set Editor from the browser

- Keep the mouse button pressed on the folder icon of tire selection field in the *Vehicle Data Set Editor*. From the drop down menu choose *Edit*:

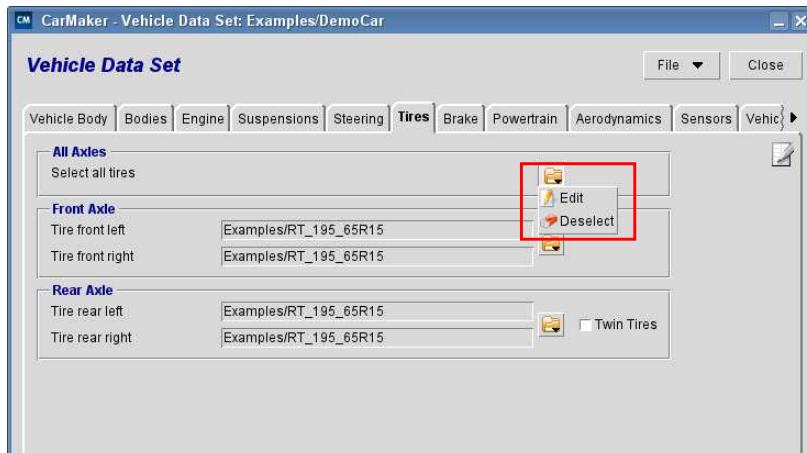
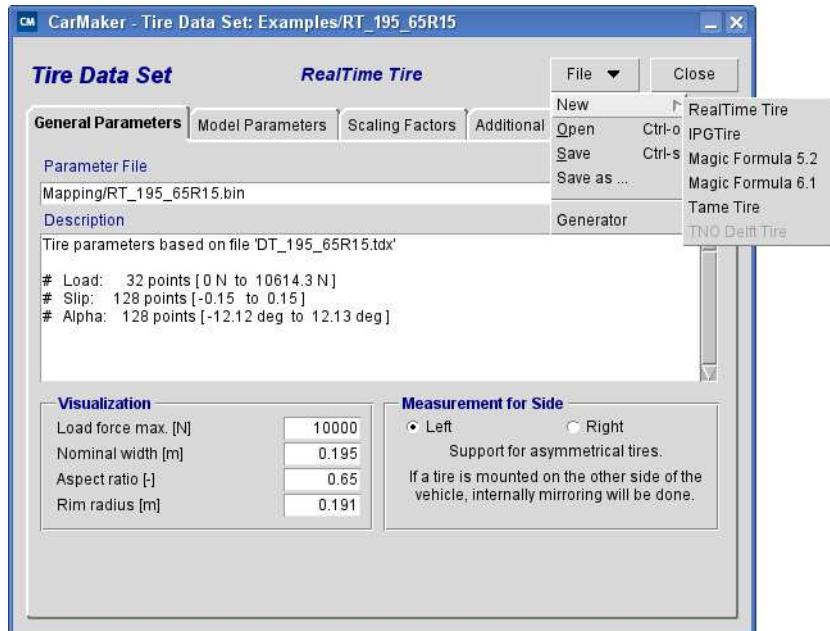


Figure 5.143: Starting the Tire Data Set Editor from the drop down menu

A new dialog pops up which holds different parameters depending on the selected tire model.

## File menu

The File menu can be used to manage existing tire data sets or to create new ones.



To create a new tire data set select the corresponding model from the list under *New*. Please find further instructions on how to proceed in the following of this chapter.

To load a new tire data set, select *Open*. Changes can be saved using *Save* and *Save As....*

Please note, that a tire data set needs to be saved anytime it was edited. Otherwise, the changes will have no effect with the start of the next simulation.

Another item is the *Tire Data Set Generator*, which can be started from here. Please find further information under [section 5.27.8 'Tire Data Set Generator' on page 350](#).



## General Parameters

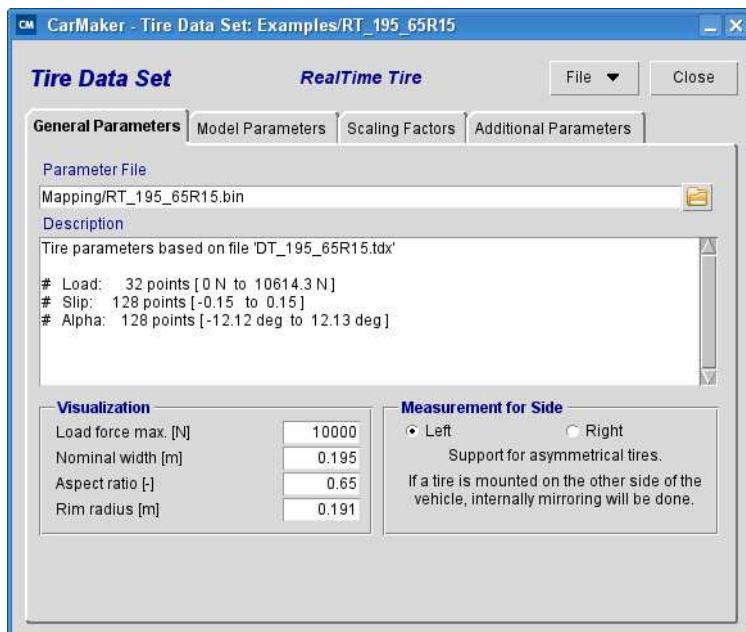


Figure 5.144: Tire Data Set Editor - General Parameters

## Visualization

All settings under Visualization do not influence the simulation result. They are used by only for the animation of the tire in IPGMovie. Thus, the following parameters are optional.

- Load Force max. (N)** Maximum animated load force of tire. Used to scale the tire force vectors in IPGMovie.
- Nominal Width (m)** Nominal width of the tire for display in IPGMovie.
- Aspect Ratio (-)** Represents the ratio between rim radius and width used for display in IPGMovie.
- Rim radius (m)** Nominal radius of rim used for IPGMovie.

## Measurement for Side

- Left/Right** Optional. Support for asymmetrical tires. If CarMaker finds an entry like "Side = left" in the tire infofile of a tire mounted on the right side of the vehicle, internally appropriate mirroring will be done automatically and vice versa. Possible values are "Left" and "Right".

## Model Parameters - RealTime Tire

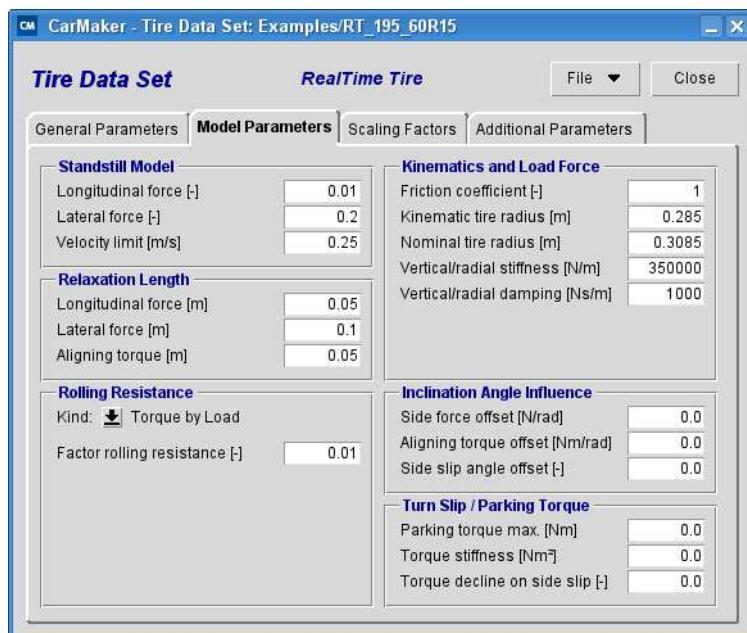


Figure 5.145: Model Parameters - RealTime Tire

### Standstill Model

- Longitudinal/ Lateral force (-)** Optional. Stiffness for standstill model. Default: 0.01, 0.2.
- Velocity limit (m/s)** Optional. Boundary for switching between standstill model and normal computation. Default: 0.25.

### Relaxation Length

<b>Longitudinal force (m)</b>	Takes into account the longitudinal tire deflection.
<b>Lateral force (m)</b>	The lateral force acting in the contact patch deflects the tire in lateral direction
<b>Aligning torque (m)</b>	The aligning torque is generated by the lateral force. By neglecting a possible dynamics of the tire offset the dynamic self aligning torque can be approximated.

### Rolling Resistance

**Kind** The following kinds of rolling resistance can be selected:

KindStr	Description
Force by Load	A force along FrW x axis, against translation based on load force.
Force by Velocity	A force along FrW x axis, against translation based on translation velocity.
Torque by Load	A torque around wheel spin axis, against rotation, based on load force.
Torque by Velocity	A torque around wheel spin axis, against rotation, based on translation velocity.
Torque by Load + Velocity	A torque around wheel spin axis, against rotation, based on load force, pressure and velocity according SAE J2452.

**Factor rolling resistance (-)** Kind dependent factor. Input is scaled by factor to get the rolling resistance output. This parameter is not required by the model Torque by Load + Velocity.

### Kinematics and Load Force

**Friction coefficient (-)** Friction conditions during tire measurement.

**Kinematic tire radius (m)** Kinematic tire radius (also known as static tire radius).

**Nominal tire radius (m)** Optional. Nominal radius of tire in non-loaded state.

**Vertical / radial stiffness (N/m)** Vertical / radial stiffness of the tire.

**Vertical / radial damping (Ns/m)** Vertical / radial damping coefficient of the tire.

### Inclination Angle Influence

The following values are optional.

**Side force offset (N/rad)** Inclination angle influence to vertical side force offset. A possible value could be -30.. Default: 0.0.

**Aligning torque offset (Nm/rad)** Inclination angle influence to vertical aligning torque offset. A possible value could be -0.9. Default: 0.0.

**Side slip angle offset (-)** Inclination angle influence to side slip angle. The current inclination angle value multiplied by this factor is added to the side slip angle value. Effects a horizontal shifting of the side force and aligning torque. A possible value could be 0.03. Default: 0.0.

### Turn Slip / Parking Torque

The following values are optional.

- Parking Torque max.(Nm)** Specifies the maximum parking torque. A possible value could be 200. Default: 0.0.
- Torque Stiffness (Nm<sup>2</sup>)** Specifies the moment stiffness against turn slip. A possible value could be 250. Default: 0.0.
- Torque decline on side slip (-)** Specifies the decline coefficient of the pure turn slip aligning moment depending on side slip angle. A possible value could be 15. Default: 0.0.

## Model Parameters - Magic Formula 5.2 and Magic Formula 6.1

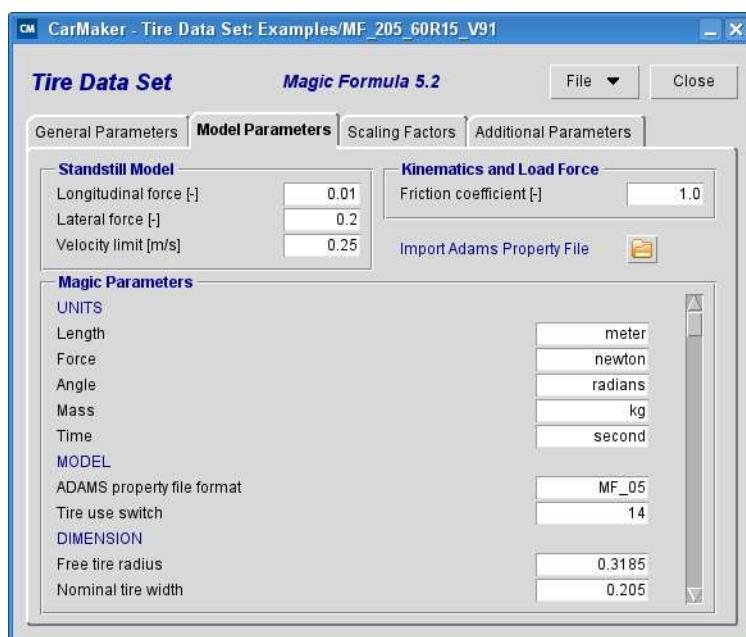


Figure 5.146: Model Parameters - Magic Formula 5.2

The "Magic Parameters" describing the Pacejka tire model can be directly edited in this tab.



If the tire parameters are available in the Adams Property File format, this file can be loaded here. All parameters are directly imported. There is no need to manually edit the parameter fields.

The list of available parameters is automatically updated when Magic formula 5.2/ 6.1 option is selected.

### Standstill Model

As the Pacejka tire model does not provide a standstill model, the standstill model implementation from the IPG Realtime Tire model is used.

- Longitudinal/  
lateral force (-)** Optional. Stiffness for standstill model. Default: 0.01, 0.2.
- Velocity limit (m/  
s)** Optional. Boundary for switching between standstill model and normal computation. Default: 0.25.

## Model Parameters - TameTire

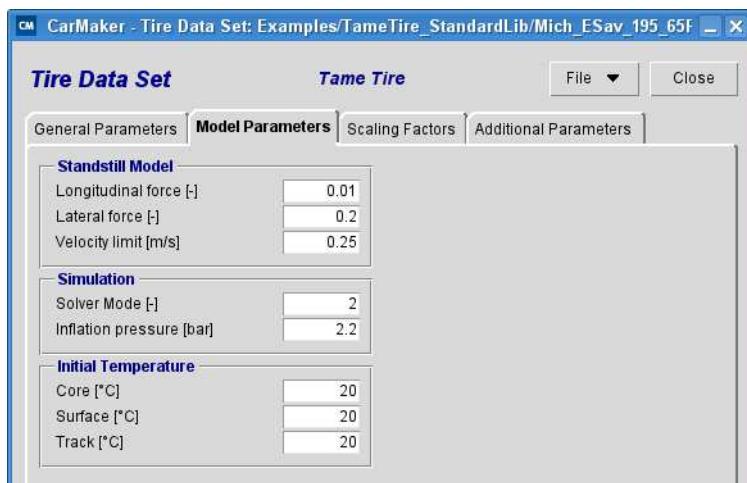


Figure 5.147: Model Parameters - TameTire

### Standstill Model

- Longitudinal/  
lateral force (-)** Optional. Stiffness for standstill model. Default: 0.01, 0.2.
- Velocity limit (m/  
s)** Optional. Boundary for switching between standstill model and normal computation. Default: 0.25.

### Simulation

- Solver Mode (-)** Specifies the solver mode of the model:
- 1: mechanical model is used (no thermal effects considered)
  - 2: Default. Thermo-mechanical model is used (thermal effects considered).
- Inflation pressure  
(bar)** Specifies the tire inflation pressure at the start of the simulation. Default: 2.0.
- Initial Temperature**
- Core (C°)** Specifies the initial core temperature of the compound. Default: 20.
- Surface (C°)** Specifies the initial surface temperature of the compound. Default: 20.
- Track (C°)** Specifies the initial track temperature. Default: 20.

## Scaling Factors - RealTime Tire

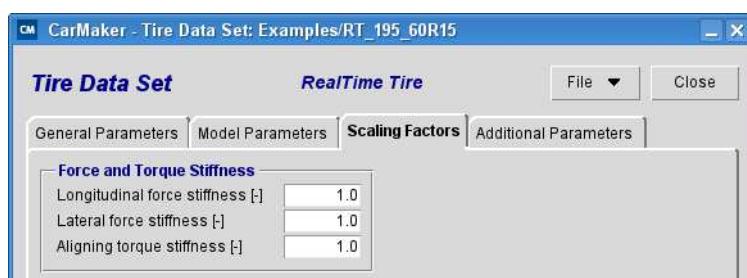


Figure 5.148: Scaling Factors - RealTime Tire

### Force and Torque Stiffness

- Longitudinal/  
lateral force  
stiffness (-)** Optional scaling factor. Default: 1.0.
- Aligning torque  
stiffness (-)** Optional scaling factor. Default: 1.0.

### Scaling Factors - Magic Formula

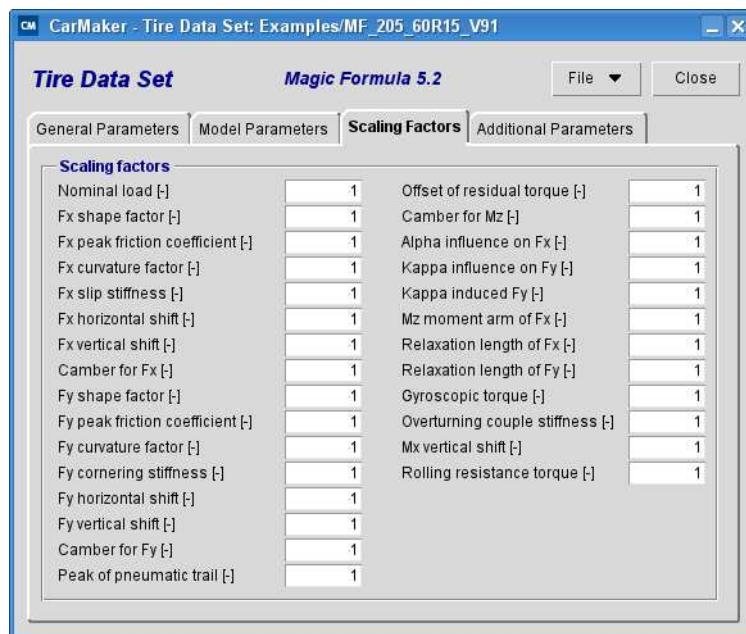


Figure 5.149: Scaling Factors - Magic Formula

All these scale factors are optional. The default value is 1.0.

## Scaling Factors - TameTire

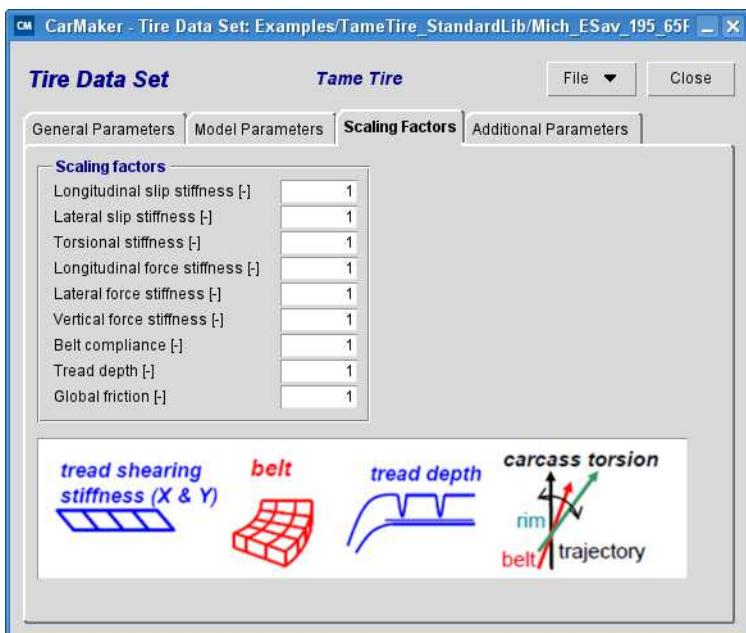


Figure 5.150: Scaling Factors - Magic Formula

All these scale factors are optional. The default value is 1.0.

## Additional Parameters

**Additional Parameters** Parameters for user specific model extensions can be written directly in the Additional Parameters field analogous to the vehicle editor.

### 5.27.4 IPGTire

#### TYDEX and IPGTire in General

The IPGTire model is based on measurements.

The basis for the tire measurements data is provided by the TYDEX format. The required files for CarMaker are generated out of a TYDEX file using the program "tireutil.exe" which is also a tool of the IPG Automotive GmbH. Thus, all you have to do to create a tire dataset is to prepare a TYDEX data file and transform it.

What is a TYDEX file? TYDEX is an abbreviation for "Tyre Data Exchange". A TYDEX file is a special format widely used in the automotive industry to export tire measurement data. It is a text file with the extension .tdx that can be opened, read and edited. The TYDEX format aims at making it easier for participating companies and institutes to exchange tire data.

Let us see how the IPGTire data set can be generated, and how it can be used.

#### IPGTire Data Set Generation Process

- |   |   |
|---|---|
| <b>1st Step<br/>Selecting the<br/>Characteristics to<br/>be Digitalized</b> | First of all, you have to obtain at least the following raw data in order to build a correct IPGTire data set: <ul style="list-style-type: none"> <li>• Longitudinal force vs. longitudinal slip</li> </ul> |
|---|---|

- Lateral force vs. slip angle
- Self aligning torque vs. slip angle

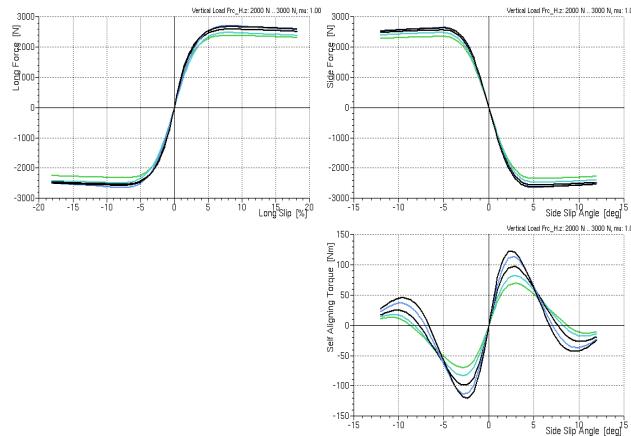


Figure 5.151: Raw data required to build a correct IPGTire data set

Without those data, the simulation results of CarMaker will not be correct enough.

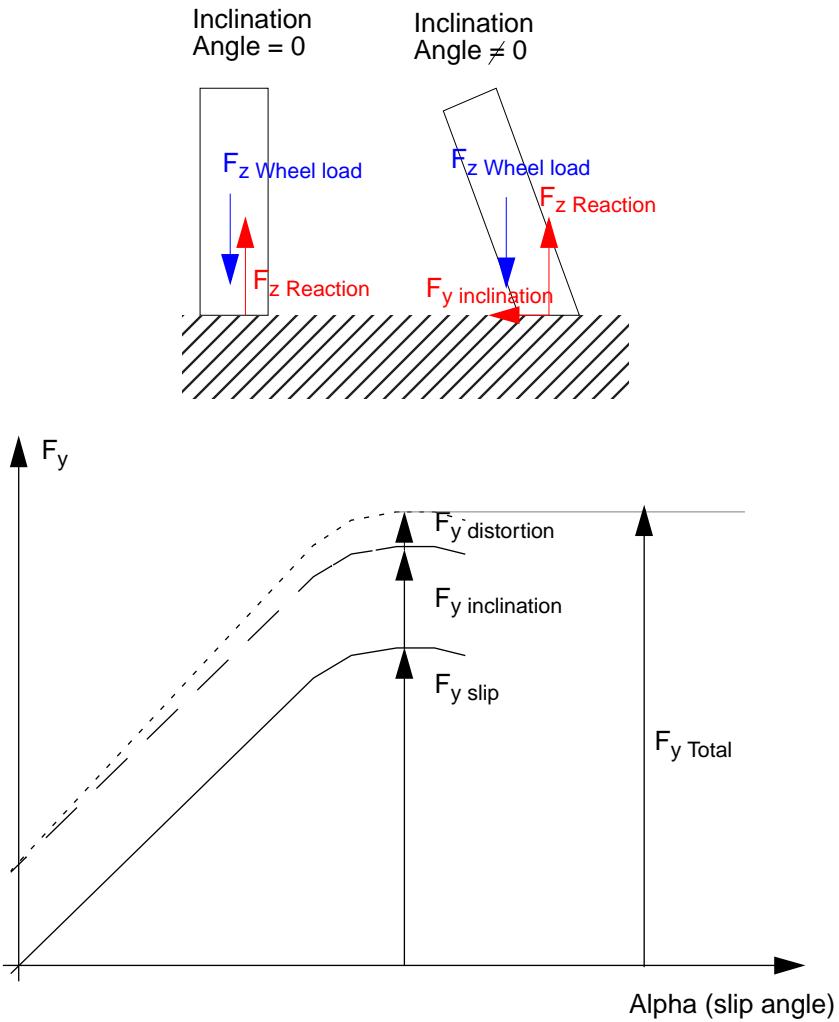


For car tires only (Motorbike tires are not concerned by the following note), you do not need to have the data mentioned above for different inclination angle values (see the definition of Inclination Angle in ISO 8855). Let us explain why.

As shown by the next picture, the lateral force applied on the contact surface area has three components (this is also hold for the self aligning torque):

- one resulting from the slip angle ( $F_y$  slip),
- another due to the inclination angle ( $F_y$  inclination),

- and a third due to the distortion of the contact surface area ( $F_y$  distortion)



$$F_y \text{ Total} = F_y \text{ slip} + F_y \text{ inclination} + F_y \text{ distortion}$$

Figure 5.152: Three components of the lateral force applied on the contact surface

In the case of a car tire, the last component can be neglected, which is not the case for a motorcycle tire.

The component due to camber angle is evaluated in CarMaker via a multiplying algorithm. If there is an inclination angle, then the side force is more important than without inclination angle. In this case, instead of taking into account the current side slip angle, the tire model considers the following calculation for the side slip angle used to calculate the side force:

$$\alpha' = \alpha + \text{InclinAngle2alpha} \times \delta$$

$\alpha'$  side slip angle used for the side force calculation

$\alpha$  actual side slip angle at the contact point of the tire with the road

$\delta$  Inclination angle of the tire at the road surface

"InclinAngle2alpha" is a multiplying factor that may be specified in the tire info file.

With this factor, the calculation uses a higher value of the side slip angle than currently applied to the tire, which enables to artificially increase the side force applied to the tire contact point.

To sum up, the consideration of the inclination angle is not done in the digitalized tire characteristics, but in the tire info file (see explanation below).

## 2nd Step Writing the Data in a TYDEX File

The measurement software installed on the tire test bed should be able to generate a TYDEX file as output directly.

As it is basically an ASCII file, it could be generated manually, too. However, the used syntax is very strict even regarding the positioning of an entry within one line. Thus, it is not recommended to generate a TYDEX file manually.

Several examples of TYDEX files (.tdx files) are available in the default CarMaker installation directory:

```
C:/IPG/hil/<ver>/Examples/TireData
```

## 3rd Step Generating the Tire Data Set for CarMaker from a TYDEX File

Although a TYDEX file contains all data required by CarMaker, it can not simply be embedded into CarMaker. The data must be transformed first: It is necessary to compress them into a binary file, because the data contained in the ASCII file is too big which will dramatically slow down the simulation.

Therefore, the IPG Automotive GmbH offers a tool called "tireutil.exe". Executing this tool, two files are generated, both needed to model a tire in CarMaker.

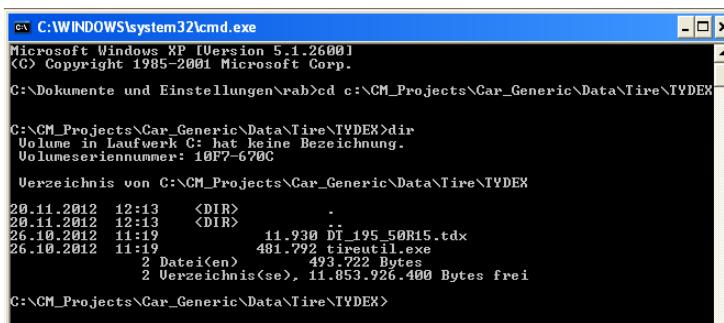
To execute this tool, it must be located in the same folders as your TYDEX file. You can find it in the installation folder "IPG", usually under: C:/IPG/hil/win32-versionNumber/bin. Copy it to your current project directory, e.g. to the new folder Car\_Generic\Data\Tire\TYDEX where you also saved the TYDEX file to be converted.

- Now, hit the "Windows Start button > Execute". To open the command line window type "cmd".
- To use "tireutil.exe" you have to browse to your project folder, e.g.:

```
C:\CM_Projects\Car_Generic\Data\Tire\TYDEX.
```

In the command line window type "C:" and hit "enter". To open the respective folder type the following line and press "enter":

```
"cd CM_Projects\Car_Generic\Data\Tire\TYDEX".
```



- Once you are located in the right directory, you have to call the "tireutil.exe" command. You have to clarify the input file (your TYDEX file, e.g. "DT\_195\_50R15.tdx") and the output files (a file without extension and a file with the .bin extension). The name of the output files can be different from the name of the input file and must be written without extension at this point:

```
tireutil -if DT_195_50R15.tdx -of TYDEX_TEST -ofbin TYDEX_TEST.bin
```

- The file generation starts (according to the size of the tydex file, it can last several seconds). Once the generation is finished, the command line is displayed:

C:\CM\_Projects\Car\_Generic\Data\Tire\TYDEX\>\_

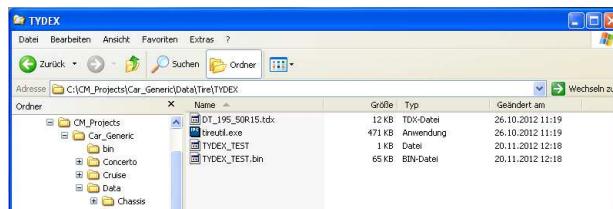
```

C:\CM_Projects\Car_Generic\Data\Tire\TYDEX\tireutil.exe -if DT_195_50R15.tdx -of
TYDEX_TEST
    Initialize tire 'DT_195_50R15.tdx'
    Requesting IPGIIIRE parameters ...
    Scanning tire ...
    Build up tire parameters
    Writing tire mapping 'TYDEX_TEST.bin'
    Writing tire 'TYDEX_TEST'

C:\CM_Projects\Car_Generic\Data\Tire\TYDEX>

```

- Now, open the Windows Explorer and browse to the folder TYDEX. Now, you can see two files respectively named TYDEX\_TEST (control file) and TYDEX\_TEST.bin (data file) in addition to the DT\_195\_50R15.tdx. Close the command line window.



#### 4th Step Using the Tire Data Sets



Once you have generated the binary file and the corresponding info file, you can use the data set in CarMaker, as you learned in the Quick Start Guide, section 'Using another Tire Parameterization'.

For this, you have to select the right infofile in the CarMaker GUI. Pay attention not to select the binary file! Use the file without file extension instead.

We recommend you to gather all binary files in a separate folder named for instance "Mapping", and to adapt the parameter "BinFName" in the info files accordingly.

#### Optional Parameters of the Info File

Some additional parameters can be defined in the info file to tune the data set or to define the aspect of the wheel in IPGMovie. They are generated automatically, but you still have the possibility to modify them.

Those parameters are described in the Reference Manual, in the section 'Tire Model "Real-Time Tire"'.

In particular, the influence of the inclination angle can be done via the optional parameter (see section '[1st Step Selecting the Characteristics to be Digitalized](#)' on page 343).



#### Asymmetrical Tire

If you digitalized an asymmetrical tire, there is the possibility to specify the current side in the info file.

For this, you have to include a parameter which defines for which side this data set is valid. If the file is loaded on the other side, then the tire characteristics will be mirrored automatically by CarMaker. The syntax for this parameter, that you can also find in the Reference Manual, is the following:

Side = left

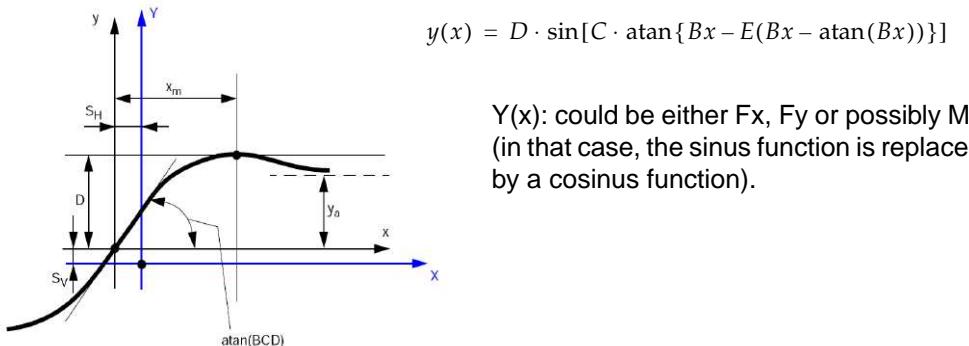
Side = right

## 5.27.5 Pacejka / MF Tire

### Magic Formula in General

The Pacejka model is based on the observation that the characteristic curves of the tires always have approximating the same shape. This shape can be described with a mathematical function, which can be tuned with a set of parameters, in order to meet the exact values found during a measurement.

The general form of the formula is:



Those parameters, called "Magic Parameters", do not have any physical meaning. They are identified with suitable programs based on tire measurements.

Additional information about the description of the Pacejka parameters can be found in the Reference Manual, section 'Tire Model "Magic Formula"'. For background information about the Pacejka model, you can use publications on Pacejka tire modeling, Magic Formula or Hans B. Pacejka's book "Tyre and Vehicle Dynamics" published in November 2002.



The source code of the Pacejka model implementation in CarMaker is also available. Thus, you can choose to compile your own version of Pacejka if you wish to (Power User only). The source code can be found in the following directory:

C:/ipg/hil/win32-<ver>/Examples/Car\_Components

Please refer to the Programmer's Guide to know how to integrate your own Pacejka code; if this is not sufficient, please contact your sales partner at IPG to let you know about the training available for this topic.

### Parameterizing a Pacejka Data Tire Set

**Version** CarMaker uses version 5.2 of Pacejka.

**Pacejka Data Set "Simple"** The simplest way to use the Pacejka parameters is to write them to an info file that you save in the ./Data/Tire folder of your project directory. For that, we recommend you to edit the tire example DT\_205\_60R15\_V91.

Once you wrote them correctly, you can load the file via the CarMaker GUI.

**Pacejka Data Set ADAMS Property Files** CarMaker offers the option to use the ADAMS property files.

In this case, we suggest to gather all ADAMS property files in a common subfolder of the "Tire" folder, e.g. "Pacejka". Please edit the file DT\_205\_60R15\_V91.tir for information, which is in the folder ./Data/Tire/Pacejka of a standard CarMaker project directory.

Then you have to create an info file that refers to the right ADAMS property file. This info file is a simple text file (ASCII format) that you can create with any simple text editor. Please edit the info file corresponding to the ADAMS property file called: DT\_205\_60R15\_V91r, which is placed in the folder ./Data/Tire of a standard CarMaker project directory.

In the info file, you have to define two parameters:

- the number of the Pacejka version. At the moment only, version 5.2 or 6.1 is supported.

```
FileIdent =CarMaker-Tire-MF52 2
```

- the path to the ADAMS property file

```
AdamsPropertyFile =Pacejka/DT_205_60R15_V91.tir
```



The file you have to load via the CarMaker GUI for an ADAMS data set is the info file, NOT the ADAMS property file itself.

Please note that the syntax of the Magic Parameters is slightly different for the ADAMS property files of the "Simple" Pacejka parameters. Please compare the syntax of the Pacejka parameters in the files ./Data/Tire/DT\_205\_60R15\_V91 and ./Data/Tire/Pacejka/DT\_205\_60R15\_V91.tir to be aware of the differences.

Starting from CarMaker 4.5, saving a the tire properties has become much more easier. When a ADAMS property file is referenced in CarMaker 4.5 and saved the data is stored along with the CarMaker Tire parameters in the same Tire info file . The format of ADAMS property file is automatically converted to info file format while storing it in the Tire info file.

```
# $Id: MF_205_60R15_V91r,v 1.2 2011/07/07 08:54:18 we ## Visualization #####
FileIdent =      CarMaker-Tire-MF52 2
                FLoadMax = 10000
                NonWidth = 0,195
                AspectRatio = 0,65
                RimRadius = 0,191
Description:
MF52 205/60R15 91V, 2,2bar
Version 23/07/2010
## MF-Tire #####
muRoad = 1,0
AdamsPropertyFile =    Pacejka/MF_205_60R15_V91.tir
Side = left
FLoadMax = 10000
# [MDI_HEADER]
MDI_Header.FILE_TYPE = tir
MDI_Header.FILE_VERSION = 2,0
MDI_Header.FILE_FORMAT = ASCII
# [UNITS]
Units.LENGTH = meter
Units.FORCE = newton
Units.ANGLE = radians
Units.MASS = kg
Units.TIME = second
# [MODEL]
Model.PROPERTY_FILE_FORMAT = MF_05
Model.USE_MODE = 14
# [DIMENSION]
Dim.UNLOADED_RADIUS = 0,3185
Dim.WIDTH = 0,205
Dim.RIM_RADIUS = 0,1905
Dim.RIM_WIDTH = 0,1924
Dim.ASPECT_RATIO = 0,60
```

Figure 5.153: Tire Info file and Adams Property file as in CarMaker 4.0 and earlier

```

## Visualization #####
FLoadMax = 10000
NomWidth = 0.195
AspectRatio = 0.65
RimRadius = 0.191

## MF-Tire #####
muRoad = 1.0

# [MDI_HEADER]
MDI_Header.FILE_TYPE = tir
MDI_Header.FILE_VERSION = 2.0
MDI_Header.FILE_FORMAT = ASCII

# [UNITS]
Units.LENGTH = meter
Units.FORCE = newton
Units.ANGLE = radians
Units.MASS = kg
Units.TIME = second

# [MODEL]
Model.PROPERTY_FILE_FORMAT = MF_05
Model.USE_MODE = 14

# [DIMENSION]
Dim.UNLOADED_RADIUS = 0.3185
Dim.WIDTH = 0.205
Dim.RIM_RADIUS = 0.1905
Dim.RIM_WIDTH = 0.1524
Dim.ASPECT_RATIO = 0.60

```

Figure 5.154: Tire Info file (Along with Adams Property file information) as in CarMaker 4.5



### Asymmetrical Tire

Please note that you have the possibility to use asymmetrical Pacejka tire data sets. For this, you have to include a parameter in the data set which defines for which side this data set is valid. If the file is loaded on the other side, the tire characteristics will be mirrored automatically by CarMaker. The syntax for this parameter, which you can also find in the Reference Manual, is the same for the "Simple" Pacejka data set as for the ADAMS property files:

```

Side = left
Side = right

```

### 5.27.6 Magic Formula 6.2 by TNO

If you would like to use the current, extended Magic Formula tire model developed by TNO, please refer to the section 'Tire Model "TNO Magic Formula"' in the Reference Manual. As the TNO tire library is licensed, you need to obtain it from TNI directly. The library can be linked to CarMaker using a SimParameter key (see Reference Manual for details).

### 5.27.7 TameTire

TameTire powered by Michelin is a detail thermo-mechanical model of a tire. It is on part of the standard CarMaker license but requires a license add on.

For detailed information about the TameTire tire model please refer to the TameTire documentation and the section 'Tire Model "Tame Tire"' in the Reference Manual.

### 5.27.8 Tire Data Set Generator

The *Tire Data Set Generator* is a tool to create generic tire data sets for a specific vehicle class and tire dimension. The algorithm is based on the Magic Formula approach and on tire characteristics defined by the European Standard Tire and Rim Technical Organization (in short *ETRTO*).

Based on the user input and the ETRTO references, the constructive parameters are determined analytically. Measurement curves defining the tire force and torque characteristics are generated with the help of Pacejka's Magic Formula.

This results are saved as measurement curves in the TYDEX format (see [section 'TYDEX and IPGTire in General'](#)). Next, the *Tire Data Set Generator* calls the *tireutil* application to convert the TYDEX file in an infofile and a binary file that can be used by CarMaker.



To analyze the generated tire characteristics and to compare them with others, use the *Model Check* ([section 6.4 'Model Check' on page 360](#)).

## Input Parameters

The *Tire Data Set Generator* is available in the *File* menu of the *Tire Data Set Editor* ([section 5.27.3 'Tire Data Set Editor' on page 336](#)):

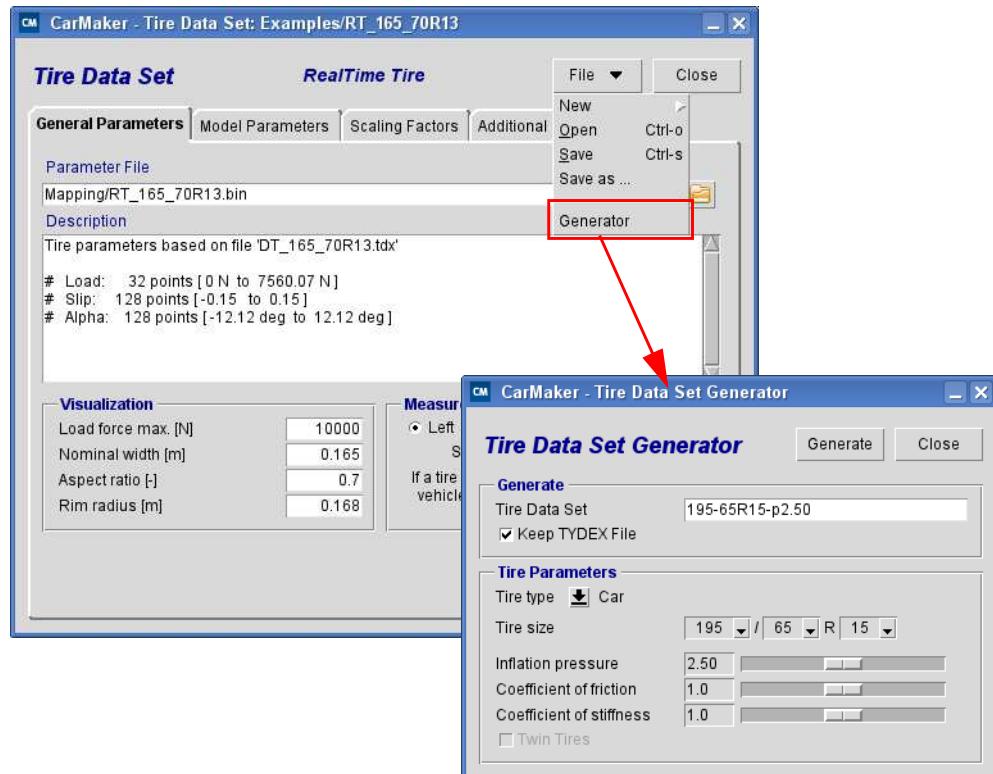


Figure 5.155: Opening the Tire Data Set Generator

**Tire Data Set** Specifies the name of the generated tire data set. Default name is tire size and inflation pressure.

**Keep TYDEX File** If activated, the TYDEX file holding the generated tire measurement curves is kept for analysis.

**Tire type** To select the vehicle class:

- *Car* for passenger cars
- *Transporter/Van* for delivery vans and light trucks
- *Truck* for heavy trucks (TruckMaker only)

**Tire size** Defines the tire dimensions: [width] / [aspect ratio] R [rim radius]

**Inflation pressure** This setting is used to modify the inflation pressure which influences the tire characteristics.

**Coefficient of friction** This coefficient directly influences the traction of the generated tire. It can be compared with the friction coefficient of the surface used by a real tire testbed.

**Coefficient of stiffness** It is a parameter to scale the stiffness of all tire characteristics ( $F_x$ ,  $F_y$ ,  $M_z$ ).

In case the generated tire should be used on an axle with twin tires, this option can be activated. It changes the load index according to the ETRTO standard and thus increases the maximum tire loads. This option is only available for vehicle classes *Transporter/Van* and *Truck*.

## Output files

To start the tire data set generation click on the *Generate* button in the top right area of the dialog. Depending on your settings, the following files will be created:

- NameOfTire

The file without file extension is the CarMaker Infofile. This file needs to be selected in the vehicle data set or the TestRun. It contains general tire parameters and a link to the binary file.

- NameOfTire.bin

This is a binary converted file. It contains the measurement curves from the TYDEX measurement file.

- NameOfTire.tdx

This file is only created with the option *Keep TYDEX File* being activated. It covers the measurement curves generated by the virtual tire test of the *Tire Data Set Generator*.

The files are saved to the "Data/Tire/Examples" folder of your CarMaker project folder. In case you chose vehicle class *Truck*, the files will be saved to the "Data/Tire/ExamplesTM" folder of your TruckMaker project folder.

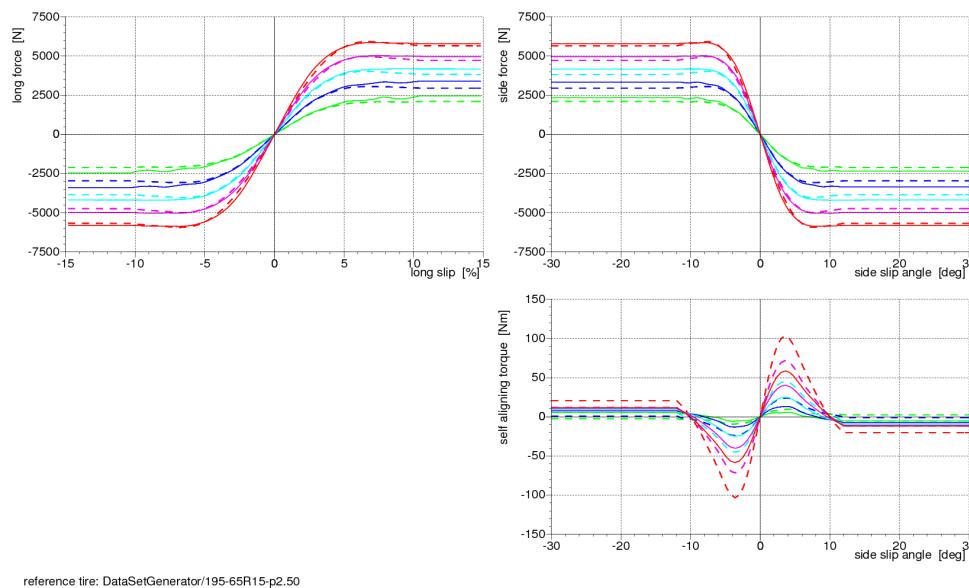


Figure 5.156: Comparison of a tire data set for a passenger car and a van created by the Tire Data Set Editor

## Chapter 6

# Simulation

## 6.1 Starting a TestRun

The whole control of the virtual vehicle environment, and with this of the CarMaker simulation, is done via the CarMaker main GUI.



Figure 6.1: Control of the virtual vehicle environment

- Start Stop** A comprehensive explanation of how to start and stop the simulation has already been given in the Quick Start Guide, Chapter 4.
- Perf.** The status of the simulation can be observed in the second red box *Simulation*. As long as you are not working in a HIL environment, you have the possibility to simulate with another performance than realtime. Various options are available to run slower or faster than realtime. At *max*, the simulation is carried out with the maximum performance available on your CPU.

**Status** The simulation can take the following status (in order of appearance):

Status	Description
Starting Application	When you start the simulation for the first time, the CarMaker GUI connects itself to the application.
idle	This is the default state when the simulation is not running. However, it does not mean the program is not working - it is simulating the vehicle's standstill.
Preparation	Here, the current vehicle parameters are read from file and the static equilibrium position is calculated.
Running	If all preparations are completed (and everything was ok), the program is really performing the simulation of the TestRun. It normally remains in this state until the TestRun ends or is aborted because of some error condition or manual intervention of the user.

Please find further information about the simulation status and the CarMaker main cycle in the Programmer's Guide, chapter 1.

**Time/Distance** The absolute time/distance covered since the start of the simulation in seconds/meters.

## 6.2 Information about Executable / Library

When working with CarMaker, you should always keep in mind that it consists of two distinct elements: the Virtual Vehicle Environment (VVE) and the CarMaker Interface Toolbox (CIT), e.g. the CarMaker graphical user interfaces (GUIs). Concretely, the VVE takes the form of an executable file (or library for CarMaker in Simulink) which contains all the models that can be run in a simulation. It is also the part that must be customized if you want to integrate your own models, define new variables that can be displayed with IPGControl, etc. The VVE and the CarMaker main GUI are started separately in most cases. With CarMaker / Office, you can select, start and stop the CarMaker executable from the Main GUI. With CarMaker / HIL, it is not possible to select the executable file.

### 6.2.1 Start and Stop the Executable / Library

With CarMaker/Office, normally the first step is to open the CarMaker Main GUI. The graphical user interface is displayed on your screen and you can do several operations like parameterizing a new TestRun, loading a vehicle file, etc. But no executable file has been started yet: The graphical interfaces run as stand alone programs without being connected to the VVE. You can notice that when you start IPGControl before starting any simulation: All the lists in the *Selection Window* are empty. They only appear once a simulation has been started. The reason is simple: the Main GUI launches the standard executable file (located in the CarMaker installation directory) if no other executable file was selected before. The message “<HostName> online” appears at the top of the Main GUI. You can also initiate this connection manually from the menu *Application*.

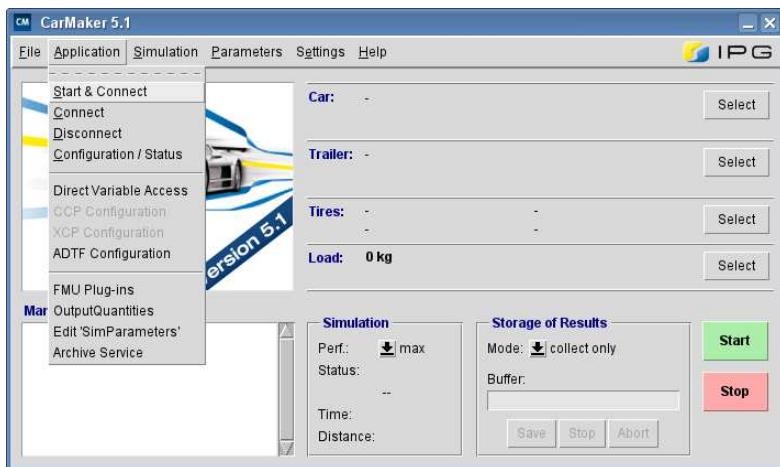


Figure 6.2: Menu “Application”

The first four items in this menu allow you to control the executable.

#### Start & Connect

By selecting this item, a new executable will be launched and some elements of the CIT (Main GUI, IPGControl, Instruments,...) will be connected to it. The loaded executable is the one specified in the configuration window (see *Configuration / Status*).

The search algorithm for the executable is as follows: First, the bin-folder of the CarMaker project folder is searched. If there is no CarMaker executable found, the bin folder of the IPG installation directory is searched.



#### Connect

The elements of the CIT look for a CarMaker executable that is running and connect to it. In this case, no new executable is started and your GUIs might not be connected to the desired executable. This option is commonly used to connect the CIT to a running HIL system. In

#### Disconnect

The GUIs will terminate their connection with the CarMaker executable.

#### Configuration / Status

When selecting this item, the configuration window for the application will be displayed on your screen. This window allows you to select the CarMaker executable to be started (and not necessarily the standard one from the installation directory), start it and make the GUIs disconnect / connect. You also have the possibility to enter a command line option.

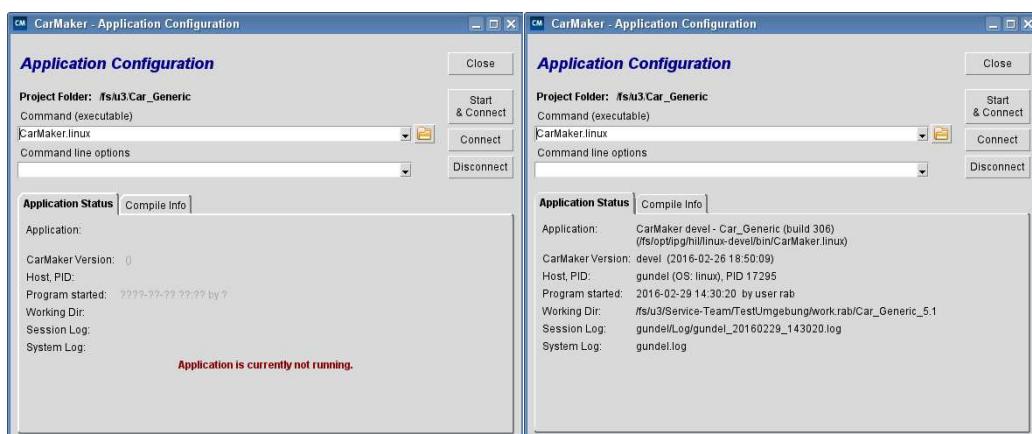


Figure 6.3: Window "Application Configuration"

When no executable is running, the tab *Application Status* is empty. A short message indicates that no application is currently running. After selecting the executable and clicking on *Start & Connect*, you can see some information about the application: name and location of the loaded executable, CarMaker version number, host, start time, working directory, log information, etc. This information allows you to clearly identify which executable has been started.



CarMaker starts per default the 32-bit executable. You can find the 64-bit CarMaker application in the installation folder under `../IPG/hil/<version>/bin`. It can be selected via the file browser available under the folder icon in [Figure 6.3](#).

With CarMaker for Simulink, the starting process is different. The standard library *libcarmakersl.dll* (Windows) or *libcarmakersl.mexglx* (Linux) that is loaded per default is also located in the installation directory. But to load another library (compiled on your own for example), you only need to place it in the *src* folder of your project directory. This library will be started automatically providing that the *src* folder appears first in the Matlab search path.

## 6.2.2 Information about the Executable and GUI

You can find additional information about the running executable in the tab *Compile Info*:

- options used for the compilation,
- libraries that were linked,
- user, date, time and computer used for the compilation.

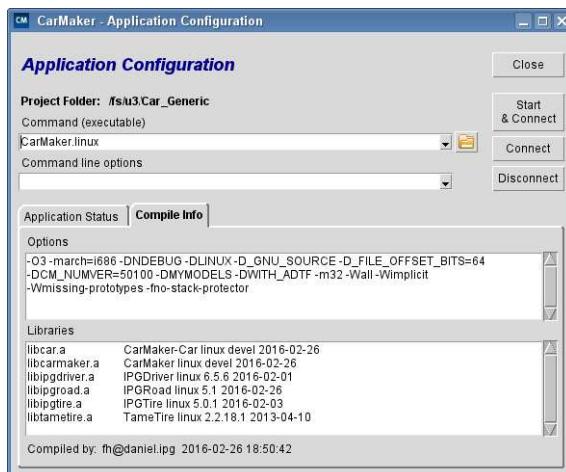


Figure 6.4: Information about executable compilation

This information allows you to identify the loaded executable with more accuracy. It is especially useful when you have compiled your own executable, to make sure that you started the right executable file.

Having a look at the linked libraries *libcar.a* and *libcarmaker.a* gives you the number of the CarMaker version you are using. This number might be different from the GUI version number that can be obtained from the menu *Help > About CarMaker*. Generally, a given CarMaker release always covers a version of the GUIs (CIT) and a version of the executable (VVE). When you start the CarMaker Main GUI and do not specify which executable is to be loaded, the executable matching the CarMaker release defined by the GUI will be loaded automatically.



This is particularly important when you update CarMaker. Be sure to start the right GUI (from the newer CarMaker release), otherwise you may still simulate with the former CarMaker version.

## 6.3 DVA: Online Manipulation of the Simulation

### 6.3.1 Usage

Direct Variable Access (DVA) enables to influence the simulation once it is already running. You can manipulate various quantities and assign a new value online. There are several ways to use the DVA mechanisms:

- DVA dialog (see below)
  - Mini Maneuver Command Language (see [section B.1.2 'Direct Variable Access Commands' on page 592](#))
  - ScriptControl (see Programmers Guide)

Using the Direct Variable Access dialog, a new value can be assigned to a quantity in a very fast and easy manner. The set button triggers the command execution. Furthermore, the DVA dialog can be used to display the current value of the selected quantity. The dialog can be found in the menu Application > Direct Variable Access:

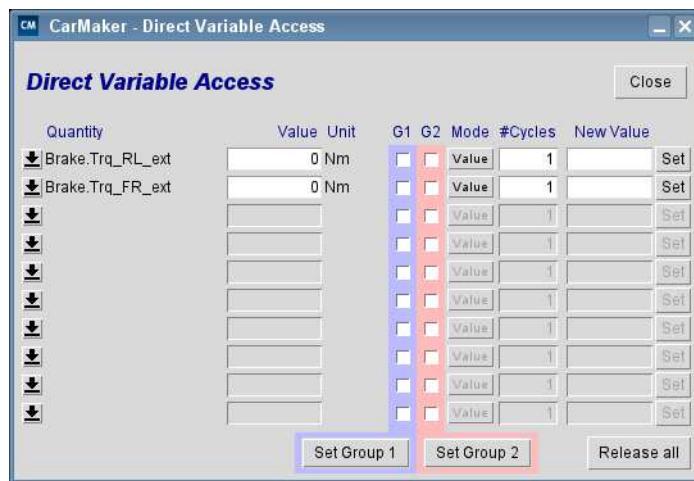


Figure 6.5: Direct Variable Access dialog

The DVA dialog enables the user to save and load various different configurations. E.g. on test bed, the same sets of quantities are usually manipulated via DVA. When saving a configuration, the quantities do not have to be selected manually again each time CarMaker is restarted. A right-click in the DVA dialog offers the several options to reset, load, save and manage configurations.

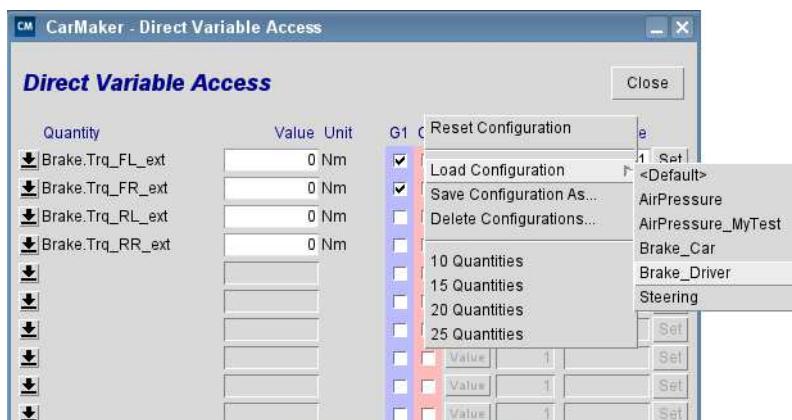


Figure 6.6: Managing different configuration in the PVA dialog

Of course, the same functionality can also be triggered by a script command which enhances the reproducibility of the test. For the script command, please check [section B.1.2 'Direct Variable Access Commands' on page 592](#). However, the difference is that the user can trigger by hand the moment when the quantity should be overwritten.

Please note, that the DVA write command can not effectively be applied on all quantities. The DVA interface only knows dedicated interaction points. To overwrite a quantity, the DVA command needs to be placed after its calculation and before it is used by other modules. With the dedicated DVA interface points it is not possible to address each quantity between calculation and usage. The quantities which can be overwritten using DVA are highlighted in the selective list. For each writable quantity, the suitable interaction point is selected automatically. Please find further information about the DVA manipulation access points in the Programmer's Guide, section "The main cycle explained".

The DVA read command can be applied on every quantity, though.

<b>Quantity</b>	In the first column <i>Quantity</i> you can select the parameter to be read or overwritten. It is the same structure as in IPGControl. However, some quantities are read-only. The quantities which can be overwritten by a DVA command are highlighted in the selective list.
<b>Value</b>	Returns the current value of the selected quantity with a sample rate of 2 Hz.
<b>G1/G2</b>	In the second column you can assign the quantity to a group. By default, two different groups are available. This is very convenient if you want to trigger the DVA write command on more than one quantity simultaneously (e.g. brake and clutch pedal).
<b>Set Values Group 1/2</b>	Here, all quantities assigned to group 1/2 (G1/G2) can be overwritten at the same time.
<b>Mode</b>	Defines the mode of the new value. The following options are available:

Table 6.1 DVA modes

Mode	Description
Abs. Value	<Val1> specifies the absolute value to overwrite the quantity.
Offset	<Val1> specifies an offset which is added to the current value of the quantity.
Factor	<Val1> specifies a factor the current value of the quantity is multiplied with.

<b># Cycles</b>	The number of cycles define for how many simulation cycles the quantity should be overwritten. One cycle has a length of 1ms at realtime simulation performance. To overwrite the quantity permanently, enter -1. However, the lack of time limitation leads to the situation, that the DVA write command is applied until a DVA reset command follows (see <i>Release all</i> ) or the whole CarMaker application is shut down. The quantity is not reset at the end of the current simulation!
-----------------	--

<b>New Value</b>	Enter the absolute value, an offset or factor to manipulate the selected quantity.
------------------	--

**set** Pressing this button, the quantity is overwritten for the duration specified at #Cycles.

<b>Release all</b>	This button releases all quantities that were overwritten.
--------------------	--



Please note: As opposed to former CarMaker versions (<4.0), the DVA access point does not have to be defined in the dialog. Each quantity which can be overwritten using DVA knows its proper interface point.

### 6.3.2 Application examples

Despite of using the DVA mechanism to overwrite certain values at a given point of time as explained before you can even build up whole submodels based on DVA.

One example would be the DVA triggered clutch model. In the vehicle data set (Parameters > Car > Powertrain > Clutch) you can select the clutch model *DVA*. At this, a disjunction of the engine and the torque transmitted to the wheels is done. The transmittable torque is not determined by the rotation speed difference of the clutch's input and output shafts and the pedal actuation. Only the torque, which is given by the user via a predefined DVA quantity (*PT.Clutch.DVA.Trq\_A2B*) is transmitted.

A DVA model also exists for the gearbox. This option can be used to define a variable gearbox transmission, e.g. a CVT gearbox. You can either manipulate the current transmission manually via the DVA GUI (parameter *PT.GearBox.DVA.i*) or refer to this parameter in your Simulink model.

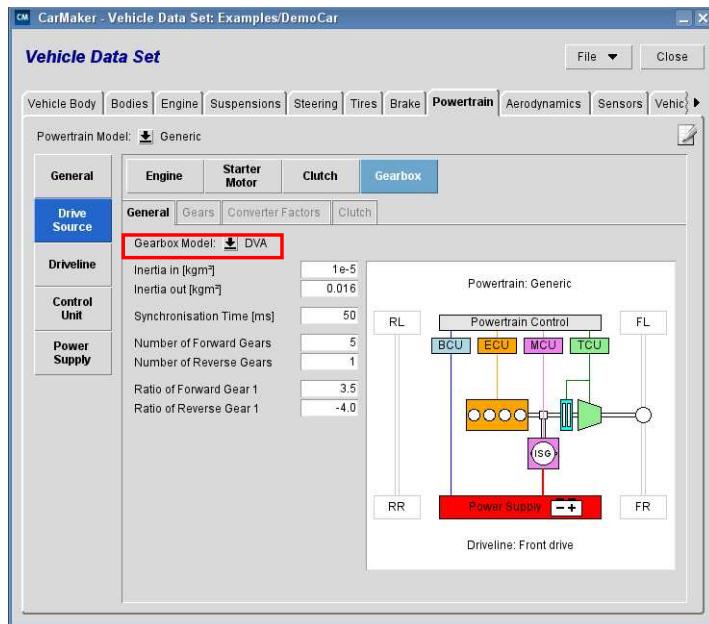


Figure 6.7: Activating the DVA clutch model

More DVA models like this exist, e.g. for the engine characteristics, the gearbox ratio or the differential. Further information on these models can be found in the Reference Manual.

## 6.4 Model Check

Before you start to simulate your TestRun, we recommend the use of the Model Check, especially if you built up your vehicle data from scratch.

### 6.4.3 Overview of the Model Check

The Model Check is a very powerful tool of CarMaker. It represents the parameterization done in CarMaker in a clear and well arranged form. Diagrams of all vehicle submodels can be generated automatically to check the data input visually and to give an overview. The huge advantage is, that diagrams compared to textual outputs are understood all over the world. They can easily be compared, show false inputs at one glance and can be used for presentations.

The Model Check dialog box can be found in the CarMaker main GUI > Simulation > Model Check. The following window pops up:

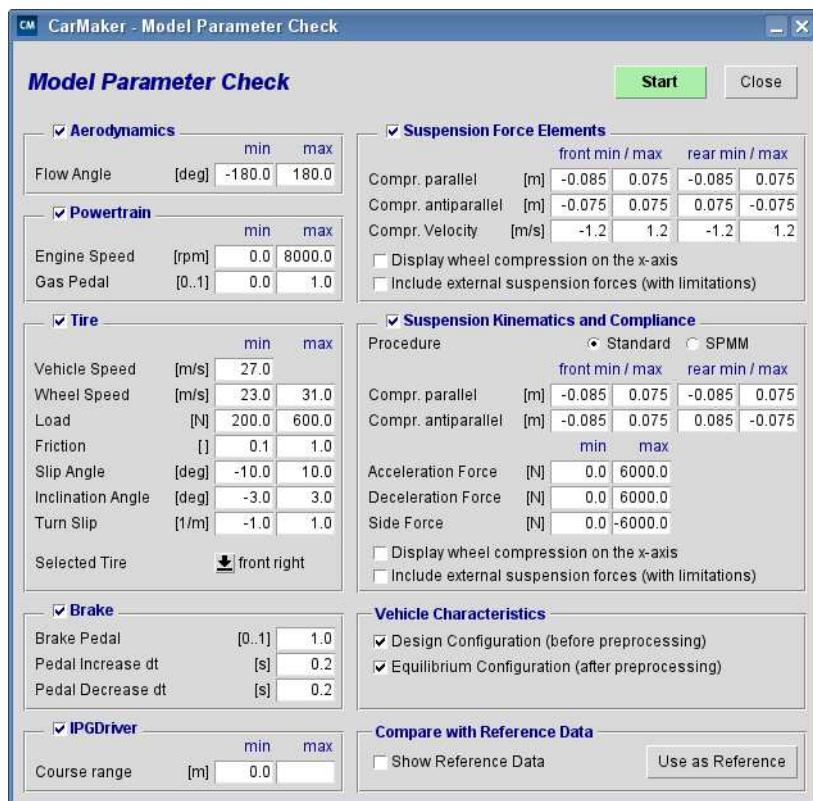


Figure 6.8: The Model Check dialog box

All subsystems of the vehicle modeling can be found in the dialog box. Additionally the parameters of the tires, IPGDriver as well as design and equilibrium configuration of the vehicle can be calculated. As the generation of the diagrams takes some time, the check boxes enable to choose only those models of interest. Each submodel needs to be given a range of the corresponding reference quantity. The diagrams generated will be kept in the limits specified here. The submodels in detail are:

- *Aerodynamics*: Specify the range of the minimum and maximum flow angle in degree.
- *Engine*: The engine speed as well as the throttle position are varied.
- *Tire*: Tire quantities are plotted versus vehicle speed, wheel speed, load, friction value, slip angle for the selected tire (front left, front right, rear left, rear right).

- *Brake*: Reference quantities are the brake pedal position and the time span the driver needs to achieve this position as well as the release time. The brake pedal actuation ranges from 0 to 1 where 1 means fully pressed. The pedal increase and decrease time is given in seconds.
- *IPGDriver*: Parameters valid for the given section of the road (in meters).
- *Suspension Force Elements*: For parallel and reciprocal compression the maximum and minimum values are given for the front and rear axle separately.
- *Suspension Kinematics and Compliance*: At parallel and reciprocal compression, also acceleration, deceleration and side forces are applied.
- *Vehicle Characteristics*: Calculates all parameters in the design and in the equilibrium configuration.
- *Compare with Reference Data*: Compares your current data with a reference TestRun.

**Start** With pressing the *Start* button at the top rear of the window, the Model Check is started. Numerous diagrams will be generated along with one text file, depending on your settings. The content will be explained in detail on the following pages.

**Show Reference Data** This option enables you to generate diagrams containing two different data sets. As Reference data any TestRun configuration can be used, containing different vehicle models or tire data sets. Activating this option, the reference data will be automatically added to the current data simulated. Both data sets are shown in the same diagram.

Note: Before you are able to use this option, the reference data has to be simulated.

**Use as Reference** Here you define which data should be used as reference data. Hitting this button starts a complete Model Check of the TestRun data currently loaded (independent on the selection you made in the Model Check GUI). The data is then stored to a buffer and keeps available for any Model Check calculation which is done afterwards. To load the reference data to the diagrams of your current Model Check, you need to toggle on the option *Show Reference Data*.



Note: Before you can perform a Model Check, a fully parameterized TestRun needs to be selected in the CarMaker GUI. This is also necessary if you only want to check data input concerning your vehicle or tires!

## Right-Click Context Menu

With a right-click in the Model Check dialog, several special settings for the Model Check parameterization and simulation are accessible.

**Default Parameters** With *Default Parameters*, the initial configuration for the Model Check settings can be loaded. This affects the parameters defining the ranges for the Model Check simulation, like flow angle for aerodynamics, engine speed and gas pedal for the powertrain etc..

**Restore Parameters** This option restores the Model Check simulation parameters from the previous run.

**Reread Parameter File** Settings, saved to the Model Check configuration file, can be loaded.

**Binary Data Files** Using this option, the Model Check output is saved to IPG erg-files (binary converted IPG result file format).

**Debug Mode** The Debug Mode writes additional output information to the CarMaker Session Log (see section 9.8 'Session Log'). This mode is only available in combination with the *Suspension Kinematics and Compliance* procedure "SPMM".

**Select/Deselect All** Select or deselect all subjects for the Model Check simulation.

**Additional Parameters** With the option Additional Parameters, a new entry field appears. Here, special parameters to reconfigure the Model Check simulation can be entered. Valid parameters are:

Table 6.2: Additional Parameters for Model Check simulation

Parameter	Description
Tire.Frc_z.Steps	Resolution of the Model Check simulation for the tire's wheel load. Default: 5
Tire.muRoad.Steps	Resolution of the Model Check simulation for the friction in tire contact point. Default: 3
Tire.vWheel.Steps	Resolution of the Model Check simulation for the tire slip. Default: 101
Tire.SlipAngle.Steps	Resolution of the Model Check simulation for the tire slip angle alpha. Default: 101
Tire.InclinAng.Steps	Resolution of the Model Check simulation for the tire inclination angle. Default: 5
Tire.TurnSlip.Steps	Resolution of the Model Check simulation for the tire turn slip. Default: 5
SPMM.Vertical.Std.P_pitch	Modifies the 'p' element of the PID pitch controller in SPMM mode. Default: 1.0e-2
SPMM.Vertical.Std.I_pitch	Modifies the 'i' element of the PID pitch controller in SPMM mode. Default: 1.0e-4
SPMM.UpdateRate	Resolution of the Model Check simulation in SPMM mode. Default: 5

## Diagram Sheets

The content of the generated diagrams will be explained on the following pages in detail.

The diagram sheets are all collected in one window. To jump between the several pages you can use the arrow buttons at the top or use the *Page Up* or *Page Down* keys on your keyboard.



The raw data for the diagrams is generated during the Model Check procedure and stored in a "ModelCheck" folder in the "SimOutput" folder inside the current project. They are general \*.erg result files and can therefore be used for postprocessing purposes as well. Please refer to [appendix 'Model Check Quantity List'](#) for a detailed Quantity description. A tire result file name description can be found there as well.

## 6.4.4 Checking the Aerodynamics

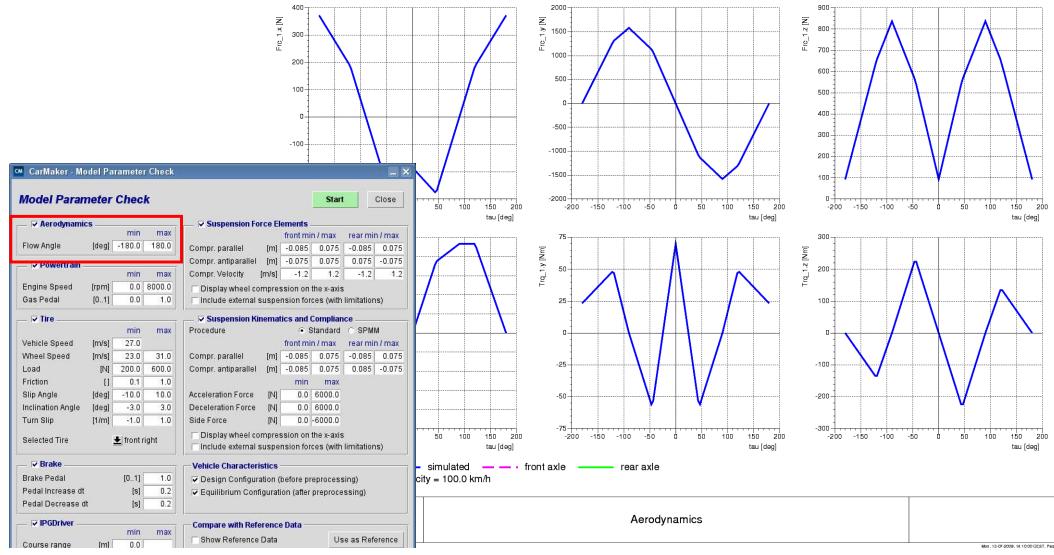


Figure 6.9: Model Check output of the Aerodynamics section

### Model Check GUI

In the Model Check GUI the reference quantity for the aerodynamics, the flow angle  $\tau$ , needs to be limited. A flow angle of zero means front wind, a negative  $\tau$  results in wind coming from the front left side.

### Output

The different aerodynamic drag coefficients against the flow angle are shown on the first page of the aerodynamics section. Followed by the resulting wind forces and torques in the three directions x, y, z. The last two pages show the drag factors and resulting wind forces for both positive and negative flow angles.

Table 6.3: Quantities displayed in the Aerodynamics section of the Model Check

Quantity	Frame	Description
cDrag [-]	-	Drag coefficient (driving direction)
cLift [-]	-	Lift coefficient (upwards / downwards)
cPitchTrq [-]	-	Pitching coefficient (around y-axis)
cRollTrq [-]	-	Rolling coefficient (around x-axis)
cSide [-]	-	Side force coefficient (lateral to the vehicle)
cYawTrq [-]	-	Yawing coefficient (around z-axis)
Frc_1.x [N]	Fr1	Drag force (positive: rearwards)
Frc_1.y [N]	Fr1	Side force (positive: to the right)
Frc_1.z [N]	Fr1	Lift force (positive: upwards)
tau [deg]	Fr1	Flow angle ( $\tau=0$ : front wind, $\tau>0$ : front left)
Trq_1.x [Nm]	Fr1	Rolling moment (positive: clockwise)
Trq_1.y [Nm]	Fr1	Pitching moment (positive: nose up)
Trq_1.z [Nm]	Fr1	Yawing moment (positive: clockwise)

More information on the meaning of the quantities displayed can be found in the Reference Manual, chapter "Aerodynamics".

## 6.4.5 Checking the Powertrain

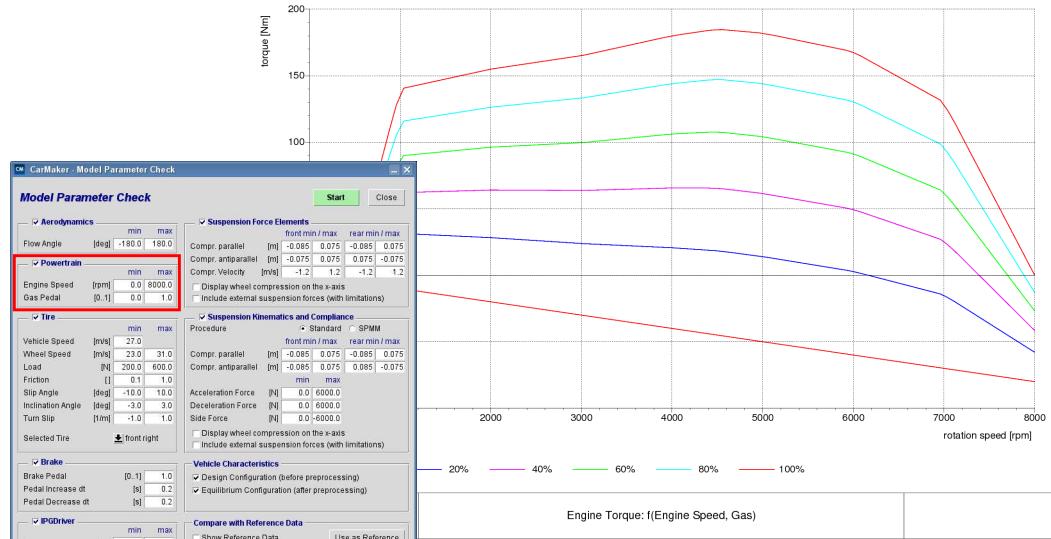


Figure 6.10: Engine output of the Model Check

### Model Check GUI

In the Model Check GUI there are two reference quantities for the engine section: the engine speed range (in rpm) needs to be limited as well as the throttle position (0 means throttle completely released, at 1.0 the throttle is fully pressed).

### Output

The diagrams of the engine section show the common engine characteristics like

- engine torque vs. engine speed at given throttle positions,
- engine power vs. engine speed at given throttle positions,
- engine torque as function of gas pedal position at various engine speeds,
- maximum engine drag force in the several gears as function of travel speed,
- maximum engine power in the several gears as function of travel speed.

Table 6.4: Quantities displayed in the Engine section of the Model Check

Quantity	Description
climbing resistance [N]	Driving resistance due to ascending slope
gas pedal [-]	Actuation of the throttle (1.0 = fully pressed)
gas pedal [%]	Actuation of the throttle (100% = fully pressed)
power [kW]	Engine power
rotation speed [rpm]	Rotation speed of the engine
torque [Nm]	Engine torque
total motive force [N]	Total engine drag force at the wheels
tractive force / power [kW]	Max. driving power at different gears
vehicle motion speed [km/h]	Velocity of the vehicle

## 6.4.6 Checking the Tire

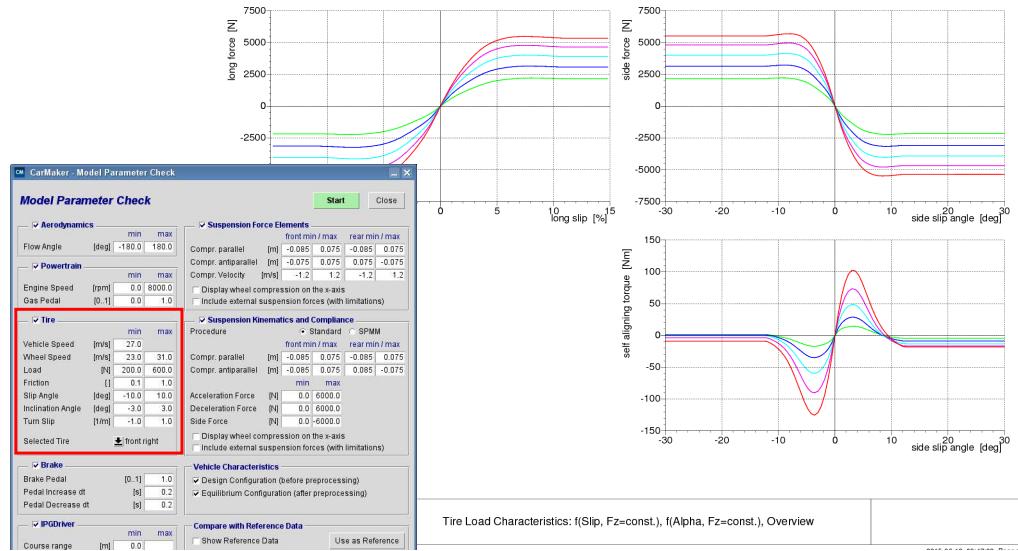


Figure 6.11: Tire characteristics at the ModelCheck

### Model Check GUI

In the Model Check GUI there are eight reference quantities for the tire section. You can define the range of interest of the vehicle and wheel speed, the minimum and maximum vertical load applied to the wheel as well as the friction, sideslip angle, inclination angle and turn slip limits. You also need to select the tire that should be proved.

### Output

The Model Check shows the common tire characteristics which makes it easy to compare the tire data used in the CarMaker model with the tire data you have from the manufacturer or from your test bench. Then you can be sure that the converted tire data in CarMaker corresponds to your measurements. The diagrams plotted represent the general tire characteristics of the one tire you selected in the Model Check GUI. The following quantities are used:

Table 6.5: Quantities displayed in the Tire section of the Model Check

Quantity	Description
friction coefficient [-]	Friction coefficient of the surface
long force [N]	Longitudinal tire force (x-direction)
(long) slip [%]	Longitudinal slip range
self aligning torque [Nm]	Self aligning torque (around z-axis)
side force [N]	Lateral tire force (y-direction)
side slip angle / alpha [deg]	Side slip angle of the tire
turn slip [1/m]	Slip of turning wheel with low velocity
vertical load / Fz [N]	Vertical wheel load applied to the tire

The sign convention is as follows:

- a positive longitudinal slip stands for traction, negative means braking.
- a positive lateral force acts on the outer side of the wheel.

Please find more information on the tire models used in CarMaker in the Reference Manual, chapter "Tire".

## 6.4.7 Checking the Brakes

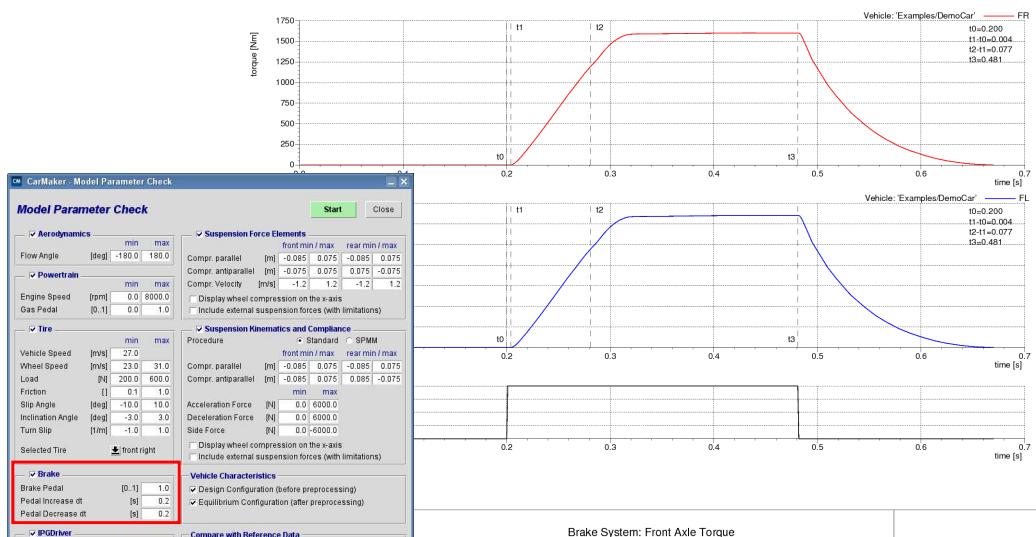


Figure 6.12: Output of the Brake System

### Model Check GUI

In the Model Check GUI there are three reference quantities for the brake section. Apart from the maximum pedal actuation the time required by the driver to fully press / release the brake pedal in seconds needs to be defined. The default value of the actuation time is zero.

### Output

The brake system is mainly characterized by the torque curve in comparison to the brake pedal actuation. The brake torque profile of each wheel brake versus time and the brake torque gradient versus time for each single wheel are plotted as well as the brake pedal actuation. Additionally to the curves some characteristic points of time are recorded. In the upper right corner of the diagrams you find information on the following time spans:

- the moment when the brake pedal is actuated
- the time passed till the pressure starts building up at the wheel brake
- the time between start of pressure built-up and 75% of maximum pressure
- the moment the driver releases the brake pedal.

Table 6.6: Quantities displayed in the Brake section of the Model Check

Quantity	Description
torque [Nm]	Brake torque at the wheels
torque gradient [Nm/s]	First derivation of the brake torque at the wheels versus time
pedal [-]	Position of the brake pedal (1 = fully pressed)
time [s]	Simulation time
t0 [s]	time when the brake pedal is actuated
t1 [s]	time when the brake pressure starts building up at the wheel
t2 [s]	time when 75% of the maximum wheel brake pressure is reached
t3 [s]	time when the brake pedal is released

## 6.4.8 Checking the Driver

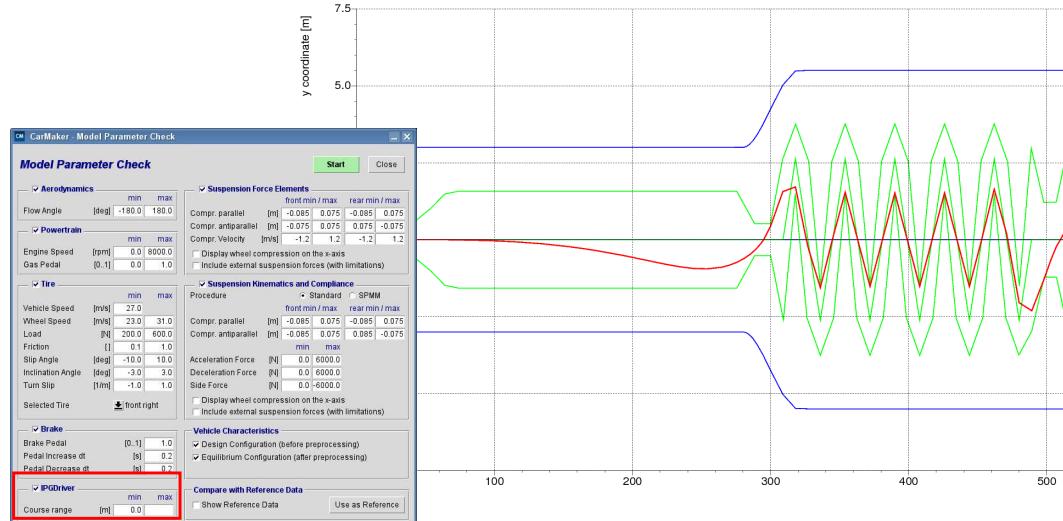


Figure 6.13: Output of the Driver

**Model Check GUI** In the Model Check GUI the reference quantity for the driver section is the course range. Here you can define if the analysis should be conducted for the whole track or only for a specific road segment.

**Output** The Driver section of the Model Check gives information about the driver's course selection. It shows you how the travelled course by the driver differs from the road length as well as velocity and acceleration profiles and some road specific parameters.

Table 6.7: Quantities displayed in the Driver section of the Model Check

Quantity	Description
acceleration [ $\text{m/s}^2$ ]	Acceleration reached by the driver
course [m]	Driver's desired course plan
curvature [1/m]	Curvature of the driver's trajectory
grad [m/m]	Gradient of the road
road width [m]	Total road width
s course [m]	Longitudinal coordinate of the driver's course
slope [m/m]	Lateral slope of the road
s road [m]	Longitudinal coordinate of the road coordinate system (following road centerline)
track width [m]	Amount of road width the driver is allowed to use
time [s]	Simulation time
velocity [m/s]	Speed reached by the driver
x / y coordinate [m]	Road coordinates in global coordinate system (Fr0)

The first diagram *Bird's Eye View* shows the driver's static desired course which is calculated at the beginning of the TestRun. Within the limits of the track the driver is allowed to move according to the driver settings (Corner Cutting Coefficient). Also displayed are the real

road's track and margin width. This is followed by a velocity and acceleration profile along the driver's course, which is also generated in the preparation phase. The curvature gives the bending of the driver's line as reciprocal of the course radius.

The *Driving Plan* shows track details against the length of the driver's planned course and the time required.

## 6.4.9 Checking the Forces Distribution in the Suspensions

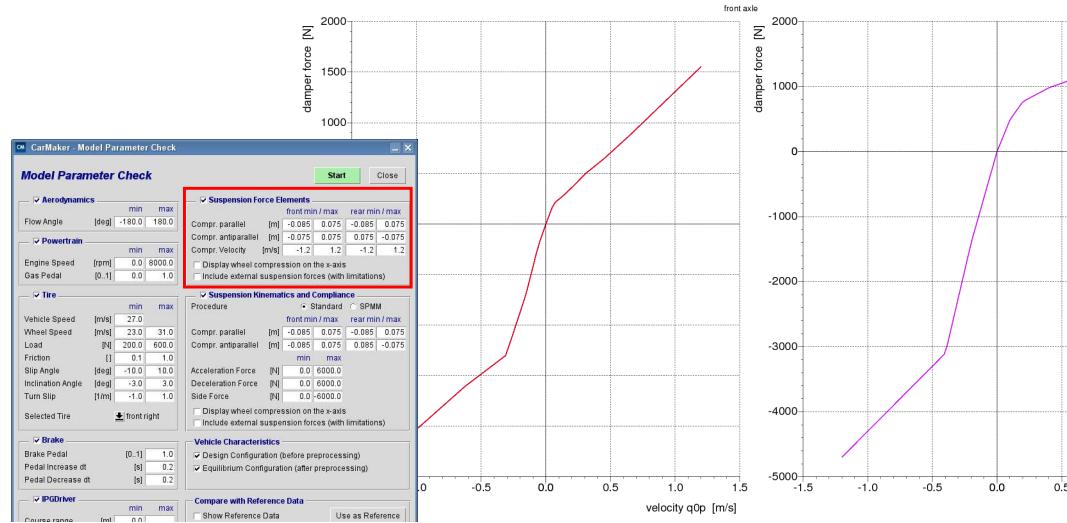


Figure 6.14: Output of the Damper Forces

### Model Check GUI

In the Model Check GUI there is the wheel compression as single reference quantity for the Suspension Force section. You define the minimum and maximum parallel and / or reciprocal compression applied to the front and rear axle separately. Optionally, the effect of external suspension forces that might have been defined in the vehicle data set can be analyzed with limitations. As long as quantities like lateral acceleration, rolling angle, velocity etc. are used in the model extension, the external forces can not be considered because Model Check is a static simulation.

### Output

This section gives an overview of the forces generated by the single suspension elements. For each element, two characteristics are shown: The component related output plots the characteristic of the single element. The effective forces display the element's characteristic on the vehicle. Front and rear axle are treated separately.

Table 6.8: Quantities displayed in the Suspension Force section of the Model Check

Quantity	Description
AxleFrc [N]	Force applied to the axle in vertical direction
buffer force [N]	Resulting force of the buffer-stop
buffer length [mm]	Buffer distance (> length of the buffer element generates a buffer force)
compression q0 [mm]	Generalized coordinate for the wheel travel in z direction
damper force [N]	Resulting force of the damper actuation
spring force [N]	Resulting force of the spring displacement

Table 6.8: Quantities displayed in the Suspension Force section of the Model Check

Quantity	Description
spring length [mm]	Absolute spring length including the displacement
stabi force [N]	Force of the stabilizer
stabi length [mm]	Length of the stabilizer
tz [mm]	Wheel travel (translation of the wheel carrier reference point in z)
velocity damper [m/s]	Velocity of the damper actuation
q0p [m/s]	Generalized velocity (derivation from q0)

## 6.4.10 Checking the Kinematics & Compliance

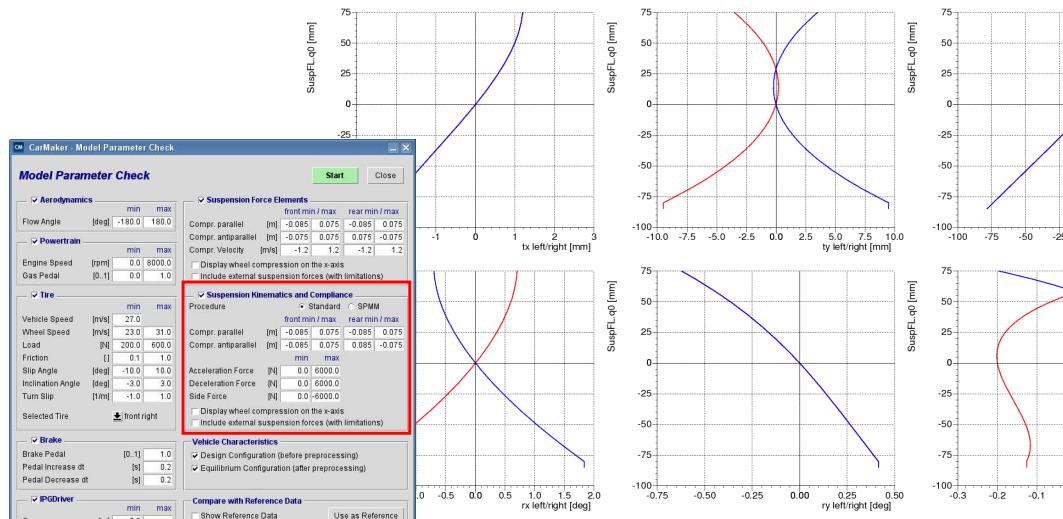


Figure 6.15: Output of the Standard Kinematics and Compliance section

### Model Check GUI

There are two ways to verify the Kinematics and Compliance (K&C) characteristics of the vehicle parameterized in CarMaker. The "Standard" mode consists of a static mode in which the K&C data that has been entered is provided to you graphically, as per the parameterization in the Suspension tab of the Vehicle Data Set editor. The "SPMM" mode consists of a simulation of the full vehicle on a virtual K&C testrig, with all the relevant results then provided graphically.

#### Standard

With the Standard mode, the wheel compression is the reference quantity for the Kinematics section. You define the minimum and maximum parallel and / or reciprocal compression applied to the front and rear axle separately.

For the compliance section the range of longitudinal and lateral tire forces has to be specified. Optionally, the effect of external suspension forces that might be defined in the vehicle data set can be analyzed with limitations. As long as quantities like lateral acceleration, rolling angle, velocity etc. are used in the model extension, the external forces can not be considered because the Standard mode of the Kinematics and Compliance Model Check is a static simulation.

#### Output

In this section you will find plots that show the kinematic and compliance characteristic of your axles under compression. The deflection and ratios of springs and dampers, change of camber, caster and toe angle and the steering influence can be analyzed against the

generalized wheel travel at parallel and reciprocal compression. Moreover, the suspension behavior under compliance shows the dependencies of the wheel positions at different wheel loads (accelerating = positive longitudinal force, braking = negative longitudinal force and cornering = lateral force). In the whole compliance section, the load is applied to the left wheel of the axle only and to the wheel center.

The characteristics for each suspension element are given for the single component (component related) and for the actuation in the vehicle suspension (effective).



Pay attention: All quantities are displayed against the generalized coordinate for the wheel hub  $q_0$ , not against the pure vertical motion of the wheel  $t_z$ !

The sign conventions are as follows:

- $rz > 0$ : toe in, anti-clockwise rotation
- $rx < 0$ : negative camber, anti-clockwise rotation
- $ry > 0$ : positive caster, anti-clockwise rotation

Please find more information on the parameters and directions in the Reference Manual, section "Suspension Kinematics and Compliance - Overview".

Table 6.9: Quantities displayed in the Standard mode Kinematics and Compliance Model Check

Quantity	Description
Accel Force [N]	Longitudinal tire force in driving direction, unsupported
Brake Force [N]	Supported longitudinal tire force against driving direction, including torque around y-axis
$drz/dqSteer$ [deg/mm]	Derivation of the toe angle with respect to the generalized steering coordinate
$drz/dSteerAng$ [deg/deg]	Derivation of the toe angle with respect to the steer angle at the wheel
$q_0$ left / right [mm]	Generalized coordinate for the wheel travel on the left / right side in z direction
rack [mm]	Travel of the steering rack
$rx, ry, rz$ [deg]	Rotation around the x- (camber), y- (caster) and z-axle (toe)
$rz$ left/right/delta [deg]	Difference of the steer angle at the two wheels of an axle
Side Force [N]	Supported lateral tire force against driving direction, including torque around x-axis
Spring / Damper / Buffer / Stabilizer length [mm]	Effective length of the spring / damper / buffer / stabilizer
Spring / Damper / Buffer / Stabilizer length rate [mm]	Change of spring / damper / buffer / stabilizer length divided by the wheel compression
SteerAng [deg]	Angle of the steering wheel
SuspFL/RL. $q_0$ [mm]	Generalized coordinate for the wheel travel on the front left/rear left wheel in z direction
$tx, ty, tz$ [mm]	Translational motion of the wheel in x, y, and z direction

## SPMM

The SPMM, unlike the static Standard, is an actual simulation which considers the full Car-Maker vehicle body and then tests it on a virtual K&C testrig with different test procedures. With the SPMM, you have the flexibility of characterizing the limits of the parameters of the various tests that are carried out on the virtual testrig.

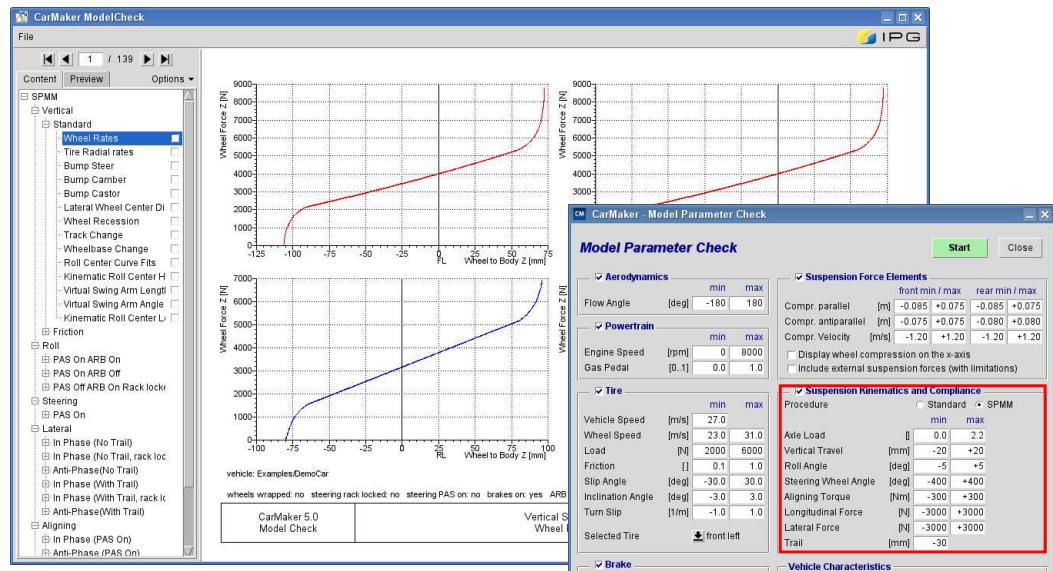


Figure 6.16: Output of the SPMM Kinematics and Compliance section

### SPMM Parameterization

The SPMM requires inputs from you regarding the limits of the inputs that are provided to the virtual K&C testrig. These inputs have been described under the various SPMM test procedures (please refer to section ['SPMM Procedures' on page 374](#)), depending on their relevance to each topic. They are also briefly described in the [Table 6.10: Description of parameter limits for the SPMM method on the Model Check GUI](#).

Table 6.10: Description of parameter limits for the SPMM method on the Model Check GUI.

GUI Parameter	Unit	Description
Axle Load	-	This is the factor that is multiplied to the force due to gravity acting on each axle. It is used to calculate the minimum and maximum force that is to be applied to the virtual wheel pads to achieve the specified force on each axle in the Vertical Test. The load is applied in the Z-axis
Axle Steer Angle	deg	-
Vertical Travel	mm	The minimum and maximum limits of the vertical travel during the Vertical Test. The displacement is applied in the Z-axis
Roll Angle	deg	The minimum and maximum limits of roll angle between which the body is cycled for the Roll Test. The angle is applied around the X-axis
Steering Wheel Angle	deg	The minimum and maximum angle within which the steering wheel is cycled during the Steering System Test.

GUI Parameter	Unit	Description
Aligning Torque	Nm	The minimum and maximum limits of the cyclic torque that is applied to the virtual wheel pads. Used for the Aligning Torque Compliance Test.
Longitudinal Force	N	The minimum and maximum force limits applied (to the tire contact point) such that the virtual wheel pads are cycled in longitudinal motion. Used for the Longitudinal Compliance Test.
Lateral Force	N	The minimum and maximum force limits applied (to the tire contact point) such that the virtual wheel pads are cycled in lateral motion. Used for the Lateral Compliance Test.
Trail	mm	Distance between the Kingpin axis at the ground to the tire contact point, when considering the side elevation. This is the point at which the forces are applied to the wheels during the Lateral Compliance Tests.

**Output** The output of all the SPMM test procedures that are carried out, are provided in a graphical format for better visualization of the K&C capabilities of the vehicle as a whole. Please take note that the terminology and the conventions used in the SPMM Model Check relate to those of an actual K&C testrig, and therefore, are different from standard CarMaker terminology and conventions. They have been elaborated upon in the [Table 6.11: Quantities displayed in the SPMM mode Kinematics and Compliance Model Check](#)

Table 6.11: Quantities displayed in the SPMM mode Kinematics and Compliance Model Check

Terminology	Unit	Description
Axle Roll Moment	Nm	The roll moments about the mid-point of the contact point positions. They are calculated using the equation: front/rear roll moment = 0.5 * (track width) * (Fzr - Fzl).
Body Bounce Displacement	mm	Vertical displacement of the testrig table centre, in ground axes. (positive is rebound)
Body Pitch Angle	deg	Angle between testrig table and ground X-axis, with respect to rotation about the Y-axis.
Body Roll Angle	deg	Angle between the table and ground Y-axis, with respect to rotation about the table X-axis. (positive is right side down).
Body Roll Moment	deg	The body roll moment is the sum of the front and the rear roll moments.
Body Ground Displacement	mm	Vertical displacement of table X axis, at front and rear axle, in ground coordinate system. (positive is bump)
Force Roll Centre	mm	The force roll centre is at the point of intersection of the resultant force vector of the left wheel and the right wheel for a given axle.
Force Roll Centre Height	mm	Vertical distance of the Force Roll Centre relative to the ground coordinate system.
Handwheel Angle	deg	Angular displacement from straight-ahead position. (positive for vehicle left turn)

Terminology	Unit	Description
Handwheel Torque	Nm	Applied torque about the axis of rotation. (positive for vehicle left turn)
Instantaneous Steering Ratio	-	The instantaneous ratio between the steering wheel angle (deg) and the steering wheel position (deg).
Kingpin Offset	mm	Distance from the Kingpin axis at ground, to the tire contact point, in front elevation on the wheel. (positive for tire contact point outside the Kingpin-ground intersection.)
Kinematic Roll Centre Height	mm	Vertical distance of the Kinematic Roll Centre relative to the ground coordinate system. The kinematic roll centre is at the point of intersection between the instantaneous roll centers of the left wheel and the right wheel for a given axle.
Scrub Radius	mm	Distance from the Kingpin axis at ground, to tire contact point, in top elevation on wheel. (positive for definition)
Static Roll Weight Transfer Coefficient (SRWTC)	-	Difference between the ratio of the roll stiffness, front over back, and the ratio of the weight transfer, front over back. If the roll stiffness ratio is greater than the static weight distribution, then the gradient of the SRWTC is positive, therefore indicating an understeering tendency (assuming all the same tires).
Steer (Ackermann)	deg	Steer angle corresponding to the pure rolling in a left turn (i.e. 100% Ackermann) (positive as for the steer angle)
Steer Angle	deg	Angle between the wheel x-axis and the ground X-axis. (positive for positive rotation about the z-axis)
Steer Wheel Position	deg	Angular displacement of the steering wheel from the straight-ahead position. (positive for a left turn)
Steer Wheel Torque	Nm	The torque applied about the axis of rotation of the steering wheel. (positive for a left turn)
Suspension Roll Angle	deg	The angle from the line joining the wheel centers of an axle to the vehicle Xv-Yv plane. (positive right side down)
Track Change	mm	Change in the lateral separation of the tire pressure centers, on each axle. (positive for increasing separation)
Trail	mm	Distance from the Kingpin axis at ground, to tire contact point, in side elevation on wheel. (positive for Kingpin ground intersection ahead of tire contact point)
Tire Contact	mm	Position of tire pressure centre in ground axes. (positive as in axis system)

Terminology	Unit	Description
Tire Compression	mm	Vertical deflection of the tire. (positive for tire compressed)
Virtual Swing Arm Angle	mm	-
Virtual Swing Arm Length	mm	-
Wheel Centre Displacement	mm	Change of the wheel centre position in ground coordinate system. (positive as in the axis system)
Wheelbase Change	mm	Change in the longitudinal separation of tire pressure centers on each side. (positive for increasing separation)
Wheel Camber	deg	Angle between the Z-axis and the wheel plane. (positive is top of the wheel inclined outwards)
Wheel Castor	deg	Angle between the projection of the steer axis on the XZ-plane and the Z-axis. (positive is top of the steering axis inclining rearwards)
Wheel Centre	mm	Wheel centre position in ground coordinate system.
Wheel to Body Z	mm	Z-component of the wheel centre displacement relative to vehicle body. (positive for bump)
Wheel Force	N	Force acting on tire in wheel pad axes. (positive as in axis system)
Wheel Toe-in	deg	Steer angle with a different sign convention. (positive for front of the wheels angled inwards)

### SPMM Procedures

A brief description of the different tests that are carried out, with their respective parameter/s for their limits are described in the following sections. Each test procedure may have different types of tests that are carried out to determine its characteristics. The constraints for each of the tests vary depending on the type of test. The different test types and constraints have been tabulated for each test procedure.

- Vertical Test** The vehicle body is cycled vertically in bounce, without any pitch. Two tests, namely, the standard and the friction tests are carried out in this procedure.

Table 6.12: Vertical Test: Description of test types

Test Type	Description
Standard	The brakes are turned on and the displacement is carried out until any one of the axles reaches its target force (force due to gravity times the Axle Load factor) corresponding to the parameterization of the Axle Load in the SPMM Model Check. The Axle Load limit that you provide is the factor that is multiplied to the force due to gravity of the axle. It has no measurement unit.

Test Type	Description
Friction	The wheel pads of the virtual test rig are displaced in equal steps between the limits provided under the Vertical Travel in the parameterization of the SPMM Model Check. The brakes are turned on during this test.

Table 6.13: Vertical Test: Description of the test constraints

Test Constraints	Description
Body	Unconstrained
Brakes	On
Steering	Handwheel fixed
Wheel pad	Lateral forces: Controlled to achieve zero. Longitudinal forces: Controlled to achieve zero
Engine	Off

**Roll Test** The vehicle body is cycled in roll, taking into consideration the maximum limits set for the Roll Angle, that has been defined in the parameterization of the SPMM Model Check. A vertical motion and pitch motion are also applied to the body, in order to maintain constant total vertical load on each axle.

Table 6.14: Roll Test: Description of test types

Test Type	Description
PAS On ARB On	The Power Assistance System of the steering and the Anti-Roll Bar are On.
PAS On ARB Off	The Power Assistance System of the steering is On and the Anti-Roll Bar is Off
PAS Off ARB On Rack locked	The Power Assistance System of the steering is Off, the Anti-Roll Bar is On and the Steering Rack is locked.

Table 6.15: Roll Test: Description of the test constraints

Test Constraints	Description
Body	Unconstrained
Brakes	On
Steering	Handwheel fixed
Wheel pad	Horizontal forces at the contact patch are controlled to zero. Free to rotate about the z-axis
Engine	On

**Steering Test** The steering wheel is cycled between the Steering Wheel Angle limits that are mentioned in the parameterization for the SPMM Model Check.

Table 6.16: Steering Test: Description of the test types

Test Type	Description
PAS On	The Power Assistance System of the steering is on.

Table 6.17: Steering System Test: Description of the test constraints

Test Constraints	Description
Body	Held Stationary
Brakes	On
Wheel pad	Horizontal forces at the contact patch are controlled to zero Free to rotate about the z-axis
Engine	On only if the PAS is On.

**Lateral Test**

The wheel pads are cycled in lateral motion between the lateral force limits that are specified in the parameterization of the SPMM Model Check. The forces are cycled in-phase and in opposite-phase. The forces are applied at the point specified by the trail in the SPMM Model Check parameterization, in relation to the centre of the tire contact pressure. The trail is considered positive if it is ahead of the centre of the tire contact pressure.

Table 6.18: Lateral Test: Description of the test types

Test Type	Description
In-Phase (0mm Trail)	A Trail of 0mm is used with the forces cycled in-phase.
In-Phase (0mm Trail, rack locked)	A Trail of 0mm and a locked steering rack are used with the forces cycled in-phase.
Anti-Phase (0mm Trail)	A Trail of 0mm is used with the forces cycled out-of-phase.
In-Phase (30mm Trail)	A Trail of 30mm is used with the forces cycled in-phase
In-Phase (30mm Trail, rack locked)	A Trail of 30mm and a locked steering rack are used with the forces cycled in-phase
Anti-Phase (30mm Trail)	A Trail of 30mm is used with the forces cycled out-of-phase.

Table 6.19: Lateral Test: Description of the test constraints

Test Constraints	Description
Body	Held Stationary
Brakes	Off
Steering	Handwheel fixed
Wheel pad	Held stationary in X-direction. Free to rotate about the Z-axis
Engine	On

**Aligning Test**

Cyclic torque in the Z-axis, i.e. Mz is applied to all four wheel pads, in various tests in-phase and opposite-phase of the left and right sides.

Table 6.20: Aligning Test: Description of the test types

Test Type	Description
In-Phase (PAS On)	The torques of both sides are cycled in phase, and the Power Assistance System of the steering is On.

Test Type	Description
Anti-Phase (PAS On)	The torques of both sides are cycled out-of-phase, and the Power Assistance System of the steering is On.
In-Phase (PAS Off, rack locked)	The torques of both sides are cycled in-phase with the Power Assistance System of the steering being off and the steering rack being locked.

Table 6.21: Aligning Test: Description of the test constraints

Test Constraints	Description
Body	Held Stationary
Brakes	On
Steering	Handwheel fixed
Wheel pad	Horizontal forces at the contact patch are controlled to zero
Engine	On only if the PAS is on.

**Longitudinal Test**

The wheel pads are cycled in the longitudinal direction, between the Longitudinal Force limits that have been parameterized in the SPMM Model Check. There are three wheel tests carried out based on varying wheel constraints, as described in [Table 6.23: Longitudinal Test: Description of the test constraints](#).

Table 6.22: Longitudinal Test: Description of the test types

Test Type	Description
Braking	The brakes are applied to emulate a braking condition.
Traction	The brakes are turned off and the transmission is locked.

Table 6.23: Longitudinal Test: Description of the test constraints

Test Constraints	Description
Body	Held Stationary
Brakes	On for Braking Off for Traction and Driven Off for Traction and Non-Driven
Steering	Handwheel fixed
Wheel pad	Lateral forces at the contact patch are controlled to zero Free to rotate about the Z-axis
Engine	On, if the brakes are applied.

**SPMM Coordinate System**

The SPMM Model Check relies on a method of coordinate systems that is specific to K&C test rigs. The different coordinate systems and their relationship to respective CarMaker coordinate systems has been elaborated in the [Table 6.24: SPMM: Description of the Coordinate Systems used in this method](#).

Table 6.24: SPMM: Description of the Coordinate Systems used in this method.

Coordinate System	Description
Ground Axis	This corresponds to the CarMaker Fr0 coordinate system. This is represented in the SPMM terminology by Xg, Yg, and Zg. For further information on the CarMaker coordinate system, please refer to the section 5.2 'CarMaker Coordinate Systems' on page 148.
Vehicle Axis	This corresponds to the CarMaker Fr1 coordinate system. This is represented in the SPMM terminology by Xv, Yv and Zv. For further information on the CarMaker coordinate system, please refer to the section 5.2 'CarMaker Coordinate Systems' on page 148
Wheel Pad Axis	This is oriented with the Ground Axis. The translational movement in the Z-direction can be compared to the generalized compression coordinate q0. For further information on the CarMaker coordinate system, please refer to the section 5.2 'CarMaker Coordinate Systems' on page 148
Wheel Axis	This corresponds to the CarMaker Fr2 coordinate system. This is represented in the SPMM terminology by Xw, Yw, and Zw. For further information on the CarMaker coordinate system, please refer to the section 5.2 'CarMaker Coordinate Systems' on page 148.

#### 6.4.11 Checking the design and static position of the vehicle

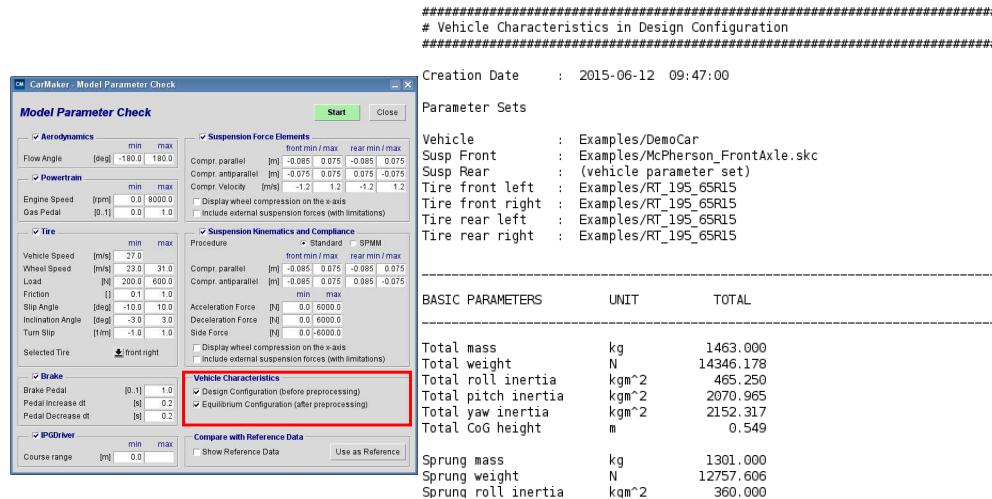


Figure 6.17: Output of the Vehicle Characteristics section

Here, the output only consists of a text file. This document is divided into two sections: the vehicle characteristics in the design configuration and in the static equilibrium configuration.

#### Design Configuration

The *Design Configuration* of the vehicle comprehends the whole parameter input done by the user in the vehicle parameterization. This includes the geometrical data as well as all other parameters such as masses, spring and damper stiffnesses etc. In most cases the design configuration is not the same as the static equilibrium configuration of the vehicle, it is a portray of the drawing sheet of the designer. E.g. the spring length and stiffness you define might not match to the vehicle's mass distribution. Not until the next step, when parameters are varied the vehicle's state matches the configuration given by the manufacturer.

In the Model Check output, the part of the *Design Configuration* gives you the possibility to check all your input data collected in one sheet.

Table 6.25: Quantities of the Vehicle Design Configuration in the Model Check

Quantity	Frame	Description
<b>Geometry in Design Configuration</b>		
RefPointDesign [m]	FrD	Translation vector of the origin of FrD from Fr1
VhclPol [m]	FrD, Fr1	Position of the point of interest (reference point for related quantities)
GenBdy1 CoM [m]	FrD, Fr1	Position of the center of mass of the generalized body (whole vehicle)
ConBdy1A / B CoM [m]	FrD, Fr1	Position of the center of mass of the connected body (rigid vehicle body incl. body, engine, trim loads and loads, B for flexible body)
TrimLoadA / B CoM [m]	FrD, Fr1	Position of the center of mass of the trim loads (B for flexible body)
LoadA / B CoM [m]	FrD, Fr1	Position of the center of mass of the trim loads (B for flexible body)
EngineA CoM [m]	FrD, Fr1	Position of the center of mass of the engine (B for flexible body)
WC <pos> [m]	FrD, Fr1	Position of the wheel center point of wheel <pos> (front left, front right, rear left, rear right)
Hitch [m]	FrD, Fr1	Position of the hitch
WheelBase (P) [m]	Fr1	Wheel base at the tire contact point with the road surface
TrackWidth (P) [m]	Fr1	Track width base at the tire contact point with the road surface
AxleFz [N]		Axle force acting on the tire contact point for front (F) and rear (R) axle, incl. distance from CoM at the front axle (F/WB) and rear axle (R/WB) to the vehicle's total center of mass
AxleMass [kg]		Axle weight for front (F) and rear (R) axle
WheelBase (WC) [m]	Fr1	Wheel base at the wheel center point
TrackWidth (WC) [m]	Fr1	Track width base at the wheel center point
Red.Masses / total [kg]		Reduced (coupling) masses at front, rear axle and center of gravity, the total weight and its inertia
<b>Masses (kg) and Inertias (kgm<sup>2</sup>), Fr1 (design configuration)</b>		
GenBdy1 CoM, I	Fr1	Weight, position of center of mass and moments of inertia of the generalized body (whole vehicle)

Table 6.25: Quantities of the Vehicle Design Configuration in the Model Check

Quantity	Frame	Description
ConBdy1 CoM, I		Weight, position of center of mass and moments of inertia of the connected body (rigid vehicle body incl. body, engine, trim loads and loads, B for flexible body)
A. / B.GenBdy1 CoM, I		Weight, position of center of mass and moments of inertia of the generalized body (B for flexible body)
A. / B.Bdy1 CoM, I		Weight, position of center of mass and moments of inertia of the connected body (B for flexible body)
A. / B.TrimLoad		Weight, position of center of mass and moments of inertia of the trim load (B for flexible body)
A. / B.Engine		Weight, position of center of mass and moments of inertia of the engine (B for flexible body)
A. / B.Load		Weight, position of center of mass and moments of inertia of the total load (B for flexible body)
A. / B.Load0..3		Weight, position of center of mass and moments of inertia of the load no. 0..3 (B for flexible body)
C<pos>+Wheel CoM, I		Weight, position of center of mass and moments of inertia of the wheel carrier and the wheel <pos> (front left, front right, rear left, rear right)
<b>PowerTrain</b>		
PowerTrain.GearBox.Kind=Manual	[ - ]	Selected gearbox kind (manual or automatic)
Ratio forward / backward gears	[ - ]	Ratio of the single gears
DriveLine.Kind = GenFront	[ - ]	Driven axle specified
DriveLine.FDiff.I_in / out	[kgm <sup>2</sup> ]	Inertia of the input / output shaft
DriveLine.FDiff.i_i2o	[ - ]	Transmission ratio of the differential
PowerTrain.Kind = Generic	[ - ]	Selected kind of powertrain
Engine.I	[kgm <sup>2</sup> ]	Inertia of the engine
Clutch.I_in / out	[kgm <sup>2</sup> ]	Inertia of the input / output plate
GearBox.I_in / out	[kgm <sup>2</sup> ]	Inertia of the input / output gearbox part
Wheel.<pos>.I	[kgm <sup>2</sup> ]	Inertia of the wheel <pos> (front left, front right, rear left, rear right)

## Equilibrium Configuration

The *Equilibrium Configuration* considers the interaction of all vehicle parameters defined by the user. The procedure is called Modify-q-equilibrium (mq-equilibrium) and it is described in the section *Finding the equilibrium state* in the Reference Manual.

Next in the calculation after the *Equilibrium Configuration* was found, is the *Start-Off Configuration*. It finally applies the loads and trim loads specified and regards non-zero velocities (e.g. when a start velocity was defined, see [section 4.5.5 'Special Maneuvers - Definition of start and end conditions' on page 98](#)). Thus, we can say, the vehicle information of this section represents the vehicle's state right before the start of the TestRun.

Table 6.26: Quantities of the Vehicle Equilibrium Configuration in the Model Check

Quantity	Frame	Description
VhclPol [m]	FrD, Fr1, Fr0	Position of the point of interest (reference point for related quantities) in the different frames
Fr1 Origin [m]	Fr0	Origin of the vehicle coordinate system Fr1 in the global system
Fr1 Roll [deg]		Roll angle around x-axis of Fr0
Fr1 Pitch [deg]		Pitch angle around y-axis of Fr0
Fr1 Yaw [deg]		Yaw angle around z-axis of Fr0
GenBdy1 [m]	FrD, Fr1, Fr0	Position of the center of mass of the generalized body (whole vehicle)
ConBdy1 [m]	FrD, Fr1, Fr0	Position of the center of mass of the connected body (rigid vehicle body incl. body, engine, trim loads and loads.)
carrier WC tx, ty, tz [m]	FrD, Fr1, Fr0	Position of the wheel center point of the wheel carrier
carrier Mnt tx, ty, tz [m]	Fr1	Translation of the mounting points compared to design position
camber [deg and min]	Fr1	Rotation angle of the wheel around x-axis
toe [deg and min]	Fr1	Rotation angle of the wheel around z-axis
caster [deg and min]	Fr1	Rotation angle of the wheel around y-axis
wheel center Fx, Fy, Fz [N]	Fr1	Forces applied on the wheel in the wheel center point
wheel road Fx, Fy, Fz [N]	FrW	Forces in the contact point of the tire on the road

## 6.5 Saving Results

Saving the data acquired during a simulation is one of the most important aspects of any testing methodology as it allows the data to be analyzed, plotted, inspected, manipulated, and otherwise used to determine exactly what took place when compared to expected results. The following description explains how to save simulation results in CarMaker.



All options you select in this dialog are in effect immediately and automatically during all of your simulations.

### 6.5.1 Starting the Saving Process

There are 3 different possibilities to start the saving process.

#### Manually

The first one is to start it manually from the *Storage of Results* area of the CarMaker main GUI.

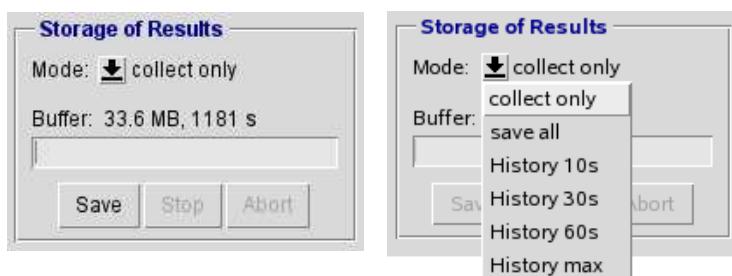


Figure 6.18: Area "Storage of Results" in the CarMaker Main GUI

In the menu *Mode*, you can select whether the simulation results are saved to the disk or not.

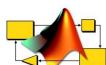
If the Storage of Results mode *collect only* is selected, the data is temporarily stored in a ring buffer without being saved to the disk automatically.

If the Storage of Results mode *save all* is selected, all data is continuously and automatically saved to the disk.



We strongly recommend to avoid the mode *save all*, as you collect a huge amount of data with every TestRun started. If the mode *collect only* is selected, the simulation results also can be saved after the simulation is finished, if you regard the data to be useful.

If one of the *History* modes is selected, the data is temporarily stored in a ring buffer. The saving process is started manually at the end of the simulation by clicking on the button *Save*. The amount of saved data depends on the history mode that has been selected. If the mode *History 10s* is selected, clicking on *Save* at the end of the simulation will save simulation results for the last 10 seconds of the simulation.



When using CarMaker for Simulink, there is no possibility to click on the *Save* button at the end of the simulation. The only way to save the simulation results with CarMaker for Simulink is by using the mode *save all*. This is so because, as soon as a simulation in CarMaker ends, Simulink unloads CarMaker (libcarmaker4sl.mex) from its memory and hence, the data-storage module of CarMaker is not available anymore to save the simulation results by clicking on *Save* button.

#### Script based

For managing simulation results in a script, please refer to the section Data Storage in the Programmer's Guide.

#### MiniManeuver Commands

Detailed information about the different Minimaneuver Commands you can find in section '[Minimaneuver Command Language](#)' on page [590](#).

The following Minimaneuver Commands are available to control the data storage:

---

**DStoreSave <TimeHistory> <TimeFuture>**

---

Start saving: store buffered data for <TimeHistory> seconds and for the following <TimeFuture> seconds.

---

**DStoreStop**

---

Stop saving: end data storing.

---

**DStoreAbort**

---

Abort saving: the result file is not kept, it is removed.

## 6.5.2 Configuring the Saving Process

### Choosing the Result folder

Per default, the saved results are stored in the *SimOutput* folder of the CarMaker project directory as follows:

```
./SimOutput/<computername>/<date>
```

where <computername> is the name of the computer on which the application is running and <date> the date when the simulation was started. It is possible to adapt this path according to your needs (see below).

The result file has per default the following name:

```
<TestRun>_<StartTime>.erg
```

where <TestRun> is the path and the name of your TestRun file, and <StartTime> the time when the simulation was started.

As an example,

```
Examples_VehicleDynamics_Braking_152010.erg
```

would be a result file of the TestRun *Braking* located in the directory *Examples/VehicleDynamics* which was started at 15:20:10.

### Quantities to be saved

In the Data Storage dialog you have the possibility to choose, which quantities calculated in your simulation you would like to save. You will find it under the menu *Application > OutputQuantities*. Every single quantity is listed in [Figure 6.19](#) (see box 1). The quantities are divided into groups and sorted alphabetically.

For the definition of all quantities see section User Accessible Quantities in the Reference Manual.

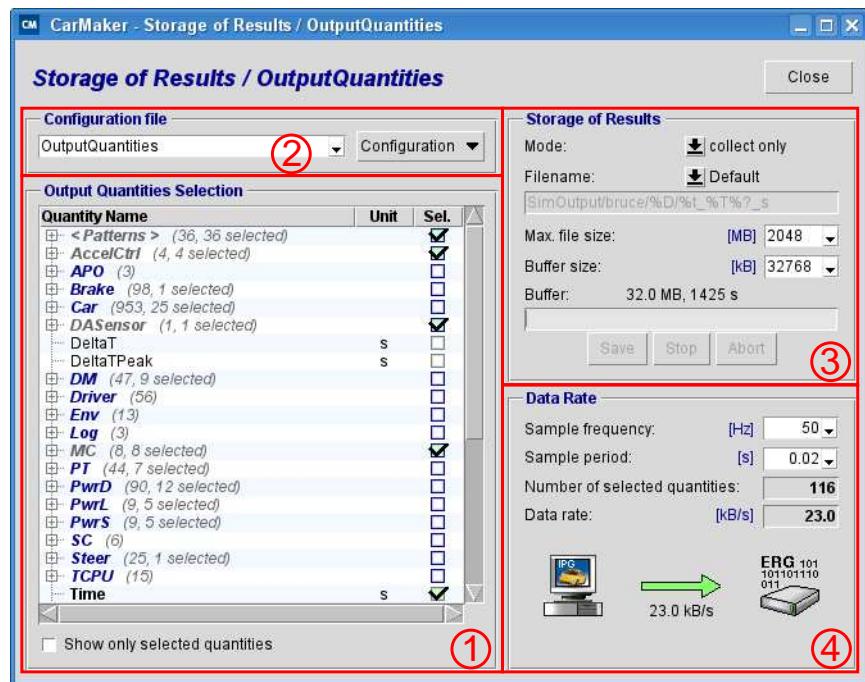


Figure 6.19: "Data Storage Browser"

Per default, a couple of quantities are selected. According to your needs, you are free to select other quantities or to deselect the predefined ones. To activate a quantity, click on the respective tick box on the right side.



If you would like to use the initial quantity selection which was active before opening the dialog, you can get it back by clicking into the list of quantities with the right mouse button. There you will find the option *Restore initial selection*.

## Configuration file

You have the possibility to save various presets of selected quantities for different types of TestRuns within one CarMaker project to a configuration file.

By clicking on the button *Configuration* in box 2 in Figure 6.19, you have the options to create a new configuration file, to load an existing one or to delete the current configuration. The default configuration *Output Quantities* can not be deleted.

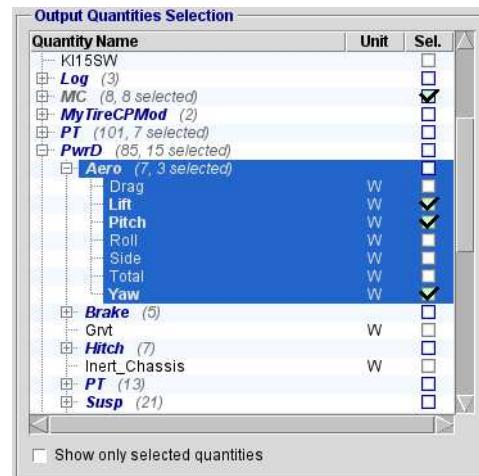


Figure 6.20: Selection of Output Quantities

If you would like to copy selected quantities to another configuration file, you have to highlight them and select the respective tick box as shown in [Figure 6.20](#). By clicking into the list of quantities with the right mouse button, you will find the option *Copy selection to...*, where you can choose the designated configuration file. The existing quantities in the configuration file will not be overwritten or deselected, this option will solely add the additional quantities into it.



With the option *Add Pattern to list* you have the possibility to add various quantities of one particular type to your selection at once. To do this, you can use a \* as a wildcard. For instance, the entry of *Car.\*.a\_o.\** will lead to all quantities, where *Car* is at the first position and *a\_o* at the third position. With this approach you can save a lot of time.

## Storage of Results

In the Storage of Results section (box 3 in [Figure 6.19](#)) you have the same possibility to configure the saving process like described in [section 6.5.1 'Starting the Saving Process' on page 382](#). But at this place, two more options are available:

- Storage Path** You can choose between the default path and an user defined path to save your designated result files to. Various macros can be used to design the output storage path. By clicking with the right mouse button into the entry field, a list with all macros and a short description of them appears.

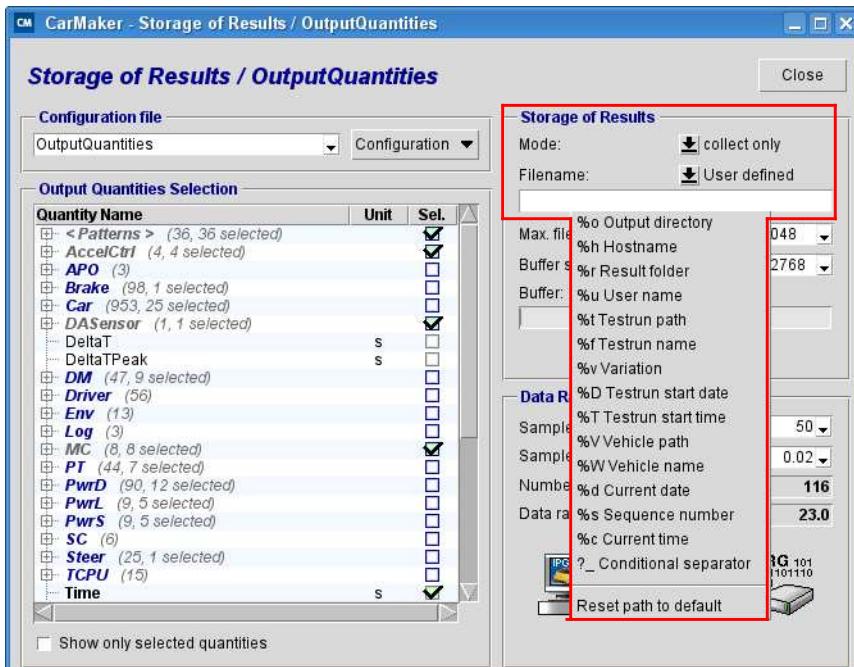


Figure 6.21: "Storage of Results"

### Example

As an example, the entry of "%f\_%T\_%W" would save your result file on top level of your CarMaker project directory with the name: "<Testrun name>\_<TestRun start time>\_<Vehicle Name>", e.g.: "Hockenheim\_142756\_DemoCar".

More detailed information on these macros can be found in the description of ScriptControl's *SetResultFName* command in section *Data Storage* in the Programmer's Guide.

### Max file size

Define a file size limit for your result file. Apart from the sizes offered in the dropdown list you can manually edit the parameter field and set a maximum file size of your choice. Please note: A file size bigger than 2GB is not supported by all postprocessing tools!

**Buffer size** You can define the maximum size of data which will be saved to the buffer of the computer. In addition you can see the time frame provided by the buffer capacity. Moreover, the buffering of the data can be saved, stopped or aborted manually through the appropriate buttons.

**Data Rate** In the *Data Rate* dialog (box 4 in [Figure 6.19](#)), you can define the sample rate of the results in Hertz (Hz) or seconds (s). By default, the sample rate is set to 50 Hz / 0.02 s.

Furthermore, the amount of the selected quantities as well as the data rate (kB/s) are shown below the save interval.

## 6.6 Postprocessing

### 6.6.1 Overview

Several possibilities exist to postprocess the simulation results of CarMaker. This task can be achieved with the following tools:

- IPGControl
- AVL Concerto (Windows only)
- Matlab
- Excel or OpenOffice
- other tools that can read text files (ASCII)

### 6.6.2 Postprocessing of the results using IPGControl



IPGControl offers the functionality of an online result management. This means that the current simulation data is provided without delay, which enables you to display diagrams directly during the simulation. By loading external result files you also have the possibility to display the results of a TestRun which has been carried out before. This is called the offline result management.

When CarMaker is opened for the first time, *Application > Start and Connect* needs to be clicked, for the quantities to be available on IPGControl. When starting IPGControl, the program looks for the running CarMaker executable. If one executable is running, IPGControl connects automatically and appears in the *Data Sets* list. If no running executable is found, the *Data Sets* list and the *Quantities* list remain empty. You can use IPGControl nonetheless in the *offline* mode. The connection with a CarMaker executable is only required for *online* viewing.

For online viewing of simulation results with IPGControl, please refer to the section "Result Management" in the Quick Start Guide.

In both cases, you first have to load your result file via the menu *File > Load File...*

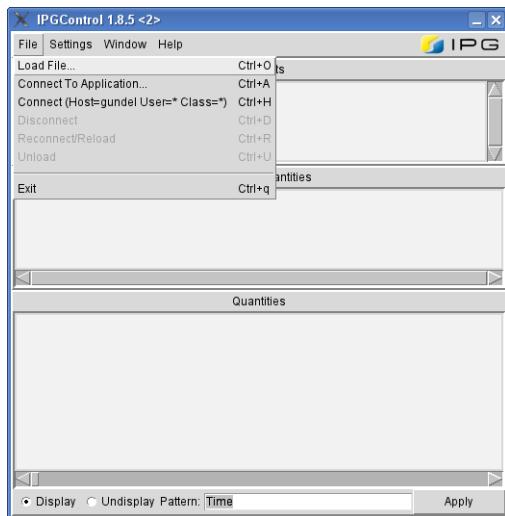


Figure 6.22: Loading a result file

After selecting the result file to be loaded, it appears in the list of available Data Sets. You may have more than one item in this list: you can simply switch from one to another by selecting it in the list. You might also want to reduce this list to only one item: you can simply unload one data set from the menu *File > Unload*. At this point, you can view and plot your results in the same way you normally do (for further explanations, please refer to the Quick Start Guide, section 'Result Management').

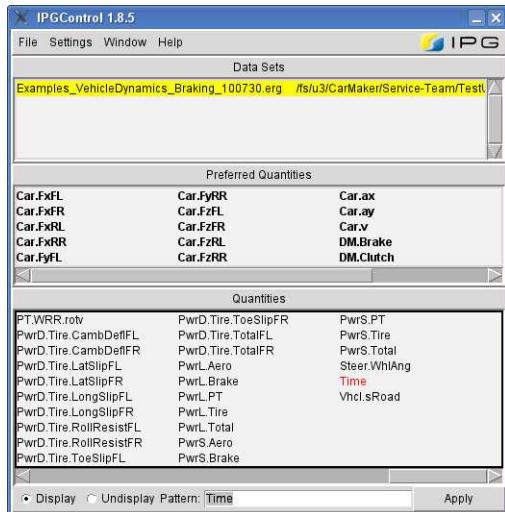


Figure 6.23: Result file loaded

When working *offline* with IPGControl, some menu items are disabled. This is the case for the sample rate configuration in the *Settings* menu. Indeed, the sample rate is the one chosen for the storage of results. It is specified in the SimParameters file by setting the value of the *DStore.dtFile* parameter. Similarly, only those quantities that were specified in the OutputQuantities configuration file will be available in the *Quantities* list (see [section 6.5.2 'Configuring the Saving Process' on page 383](#)).

Loading more than one result file is possible: each loaded file will then be listed in the *Data Sets* list of the selection window. But only one result file will be active: it is the one overlined with yellow. When opening a new *Data Window*, you can plot and view results in a diagram for the active result file. It is then possible to activate another file, open a new *Data Window* and plot new diagrams without losing the diagrams related to the previously activated file.

### 6.6.3 Postprocessing using CONCERTO

CONCERTO is an efficient and powerful postprocessing tool from the AVL List GmbH. In the following is explained how to use CONCERTO in the CarMaker environment. This chapter will describe how CONCERTO can be used as a postprocessing tool for CarMaker, to get more information about CONCERTO please consult the CONCERTO manuals.

#### The CONCERTO work environment

To use CONCERTO as postprocessing tool for CarMaker it is necessary to copy the CONCERTO environment into the CarMaker project directory. This can easily be done by activating the Concerto Work Environment check button after choosing File > Create / Update Project ... .

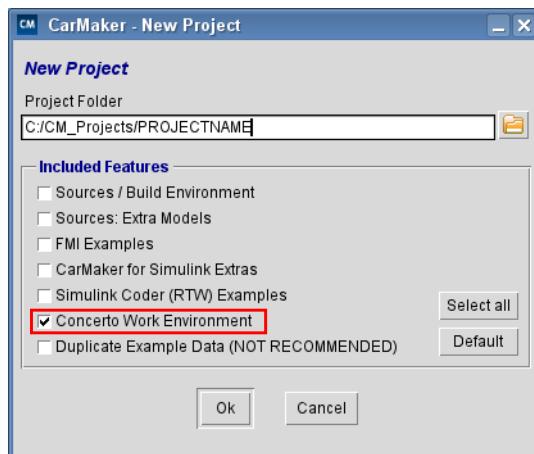


Figure 6.24: Adding the CONCERTO Work Environment to your CarMaker project folder.

After copying the CONCERTO environment there will be a new directory Concerto inside the CarMaker project folder. Within this directory a set of subdirectories can be found. The Layouts directory contains several predefined layout files which allow a fast analysis of simulation results.

#### How to start CONCERTO within the CarMaker environment

There are three possible ways to start CONCERTO from the CarMaker environment. In each case the CONCERTO working environment is set to <CarMaker Project>/Concerto.

- File > Concerto > Start Concerto: will start CONCERTO (see [Figure 6.25](#))
- File > Concerto > <layoutgroup> > <layout>: start with loading a layout and the latest simulation results found in SimOutput (see [Figure 6.25](#))
- Use ScriptControl Commands (see the Programmer's Guide for more information)

Starting CONCERTO automatically generates two files: tmp\_ConcStart.csf and WorkEnvironmentData.dvx. The tmp\_ConcStart.csf is a CONCERTO script which will be executed right after start of CONCERTO. It will be deleted from CONCERTO automatically. This script causes CONCERTO to change the WorkEnvironment to <CarMaker Project>/Concerto, and loads layout and result file. The file WorkEnvironmentData.dvx contains a list of available DataSources inside the SimOutput directory. CarMaker will start CONCERTO with a certain instance ID. Another start of CONCERTO from within CarMaker does not start

a new CONCERTO instance but the already existing instance will execute the newly created tmp\_ConcStart.csf and CONCERTO will read the also newly created WorkEnvironmentData.dxf file again to update DataSources.

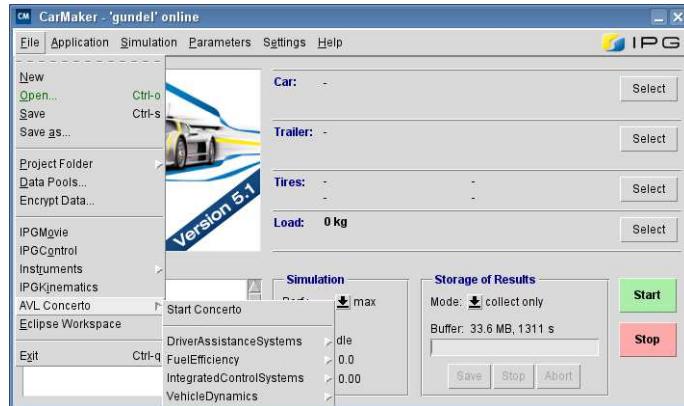


Figure 6.25: Starting CONCERTO

### Predefined CONCERTO layouts

As described above you can start CONCERTO using a predefined layout. The layouts are grouped in topics and the diagrams display the quantities of interest for the certain tests. These layouts should serve as a basis for data analysis giving you an idea of the capabilities of CONCERTO.

Every layout is optimized for a specific TestRun, but it can be applied to any simulation result. Please note, that in this case some quantities might not be available or that the scaling of the axis might have to be changed.

- Driver Assistance Systems

Here you can find the layout called ACC. It contains diagrams giving information about the longitudinal and lateral motion of the vehicle and the obstacles, the desired distance and speed.

It was designed for the use in the TestRun "Examples/DriverAssistanceSystems/ACCAndPreCrash/ColumnDriving\_SheerIn".

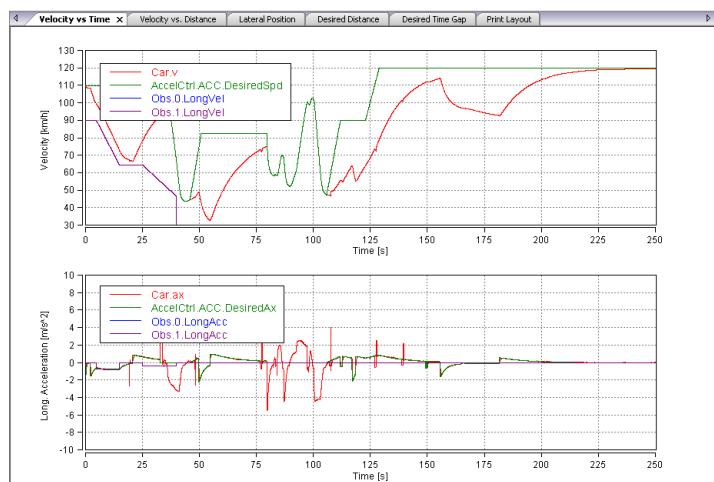


Figure 6.26: Predefined layouts for Driver Assistance Systems tests

- Integrated Control Systems

This category contains the layout called "ABS\_Braking\_musplit". Besides the velocity and distance covered by the vehicle it shows you the recorded signals of wheel brake pressure, longitudinal slip, steering wheel angle, yaw rate and lateral acceleration.



This layout was created for the use in the TestRun "Examples/IntegratedControlSystems/ABS\_Braking\_musplit".

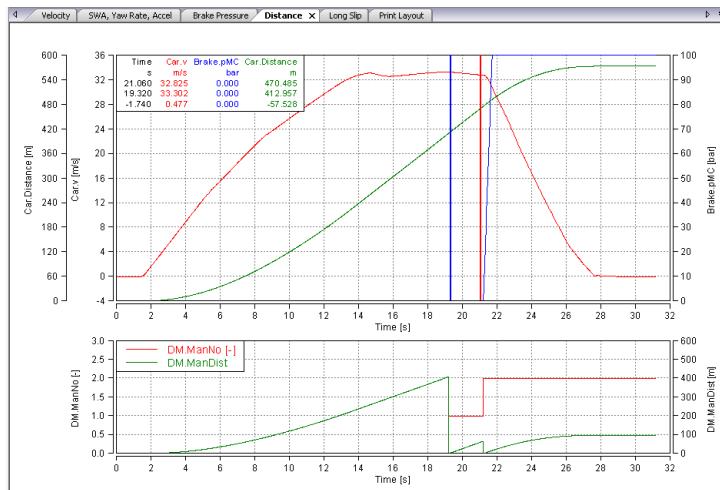


Figure 6.27: Predefined layouts for Integrated Control Systems tests

- Fuel Efficiency

The absolute, current and average fuel consumption can be analyzed here as well as the power loss and power store of different vehicle units.



The layout is called "FuelEfficiency" and should be used in combination with the TestRun "Examples/FuelEfficiency/DriveCycle\_NEDC".

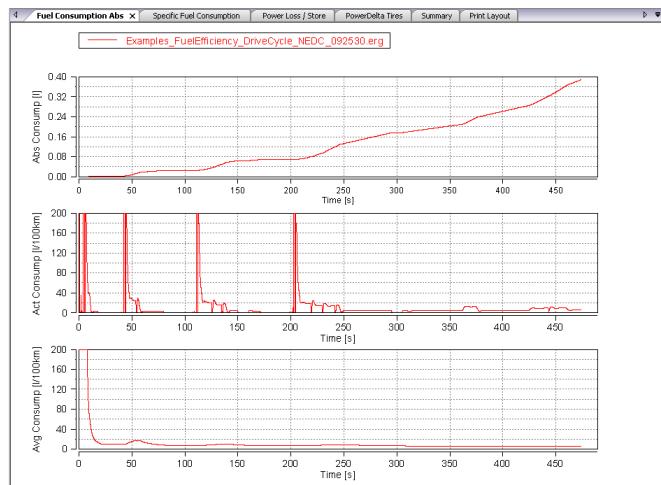


Figure 6.28: Predefined layouts for Fuel Efficiency tests

- Vehicle Dynamics

In the vehicle dynamics, templates for the most common tests Braking, Lane Change and SteadyState Circle 100m are provided. All TestRuns can be found in the "Examples/VehicleDynamics/" section. The characteristic of the vehicle can be evaluated using the standard quantities for this tests.

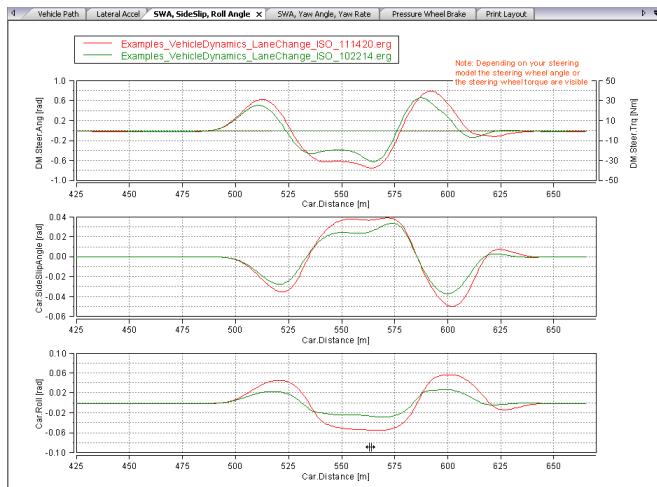


Figure 6.29: Predefined layouts for Vehicle Dynamics tests

#### 6.6.4 Postprocessing with MATLAB

One function of the Matlab support package for CarMaker offers the possibility to export CarMaker simulation results into the Matlab workspace. Once available in the workspace, the simulation data can be used and manipulated in every conceivable way, e.g. for post-processing and plotting.

##### Starting Matlab

Before calling any function of the Matlab support package for CarMaker, it is necessary to configure the Matlab search path. Otherwise, the use of a function will cause a Matlab error to occur. To configure it easily, you just have to start Matlab from the "src\_cm4sl" folder of your project directory. This will automatically load the Matlab script "cmenv" which configures the Matlab search path. When processed successfully, a few lines should be displayed in the Matlab console. If it is not the case, you can still execute the script manually by typing the following command in the Matlab console:

```
>> cmenv
```

##### Invoking cmread

The function cmread makes it possible to export simulation results into Matlab. Basically it works as follows:

- cmread lets you specify a path, either to a directory or to a file. In case a directory is specified, you may select a file in a file browser window. If no path is given, the file browser will be executed in the current Matlab working directory.
- The return value of cmread must be assigned to a workspace variable. The result is a Matlab structure with one member for each loaded variable. Thus, several result files may be loaded into the workspace (e.g. for comparison), the data of each result file being stored in a different workspace variable.
- By specifying one or more patterns, it is possible to select only a subset of all variables contained in a results file. By default, all variables are loaded into the workspace.

The next section contains some examples that show how cmread works. For further details see the cmread online help page in Matlab, by typing

```
>> help cmread
```

at the Matlab prompt.

### Example Use

#### Loading a result file

- Open a file browser window, select the result and assign the file data to the workspace variable a:

```
>> a = cmread;
```

- Open a file browser window with the contents of your CarMaker result directory, select the result file and assign the data to the workspace variable a:

```
>> a = cmread('../SimOutput/<myhost>');
```

You may have to replace the name <myhost> with the hostname of the computer you are using.

- Load two different simulation result files into the workspace:

```
>> a = cmread('SpecialManoeuver1.erg');
>> b = cmread('SpecialManoeuver2.erg');
```

#### Working with the loaded data

- Display the contents of variable a, which contains the loaded data:

```
>> disp(a)
Car_CFL_rx:[1x1 struct]
Car_CFL_ry:[1x1 struct]
Car_CFL_rz:[1x1 struct]
...
Time [1x1 struct]
```

Please note the difference between the variable names in the result file and the names of the structure members. cmread automatically renames variables that do not match the syntax of Matlab's data types. This includes changing "." to "\_" and shortening long names if necessary.

- Display the contents of the Car.vx variable:

```
>> disp(a.Car_vx)
unit: 'm/s'
nstates: 0
data: [1x10144 double]
```

- Show a speed-over-time diagram, plot the course of the car:

```
>> plot(a.Time.data, a.Car_vx.data)
>> plot(a.Car_Frl_tx.data, a.Car_Frl_ty.data)
```

#### Loading only a subset of variables from a result file

- Open a file browser window showing the current directory, select the result file and assign all DrivMan and PT variables as well as the Time to the workspace variable a:

```
>> a = cmread('.', 'Time', 'DM*', 'PT*');
```

Function arguments following the specified directory "." maybe variable names as well as patterns of names. cmread may be invoked with any number of name/pattern arguments. Patterns follow the syntax that normally can be used for filenames. The special characters most often used for patterns are listed below:

- \* matches any string including the empty string
- ? matches any single character
- [...] matches any of the characters enclosed between the brackets

## 6.6.5 Data Conversion (ASCII, CSV, XLS)

### Plotting simulation data with IPGControl

Before converting your data into the desired format, you have to load the result file in IPGControl and plot the variables you would like to export in a diagram. Note that only the variables that are plotted in the diagram will be exported.

For further explanations about how loading and plotting simulation results, please refer to section 6.6.2 'Postprocessing of the results using IPGControl' on page 386.

### Exporting simulation data to the desired format

Once all the data you want to export is plotted in a diagram, you just need to right-click anywhere in the diagram area and select *Export to File* in the menu.

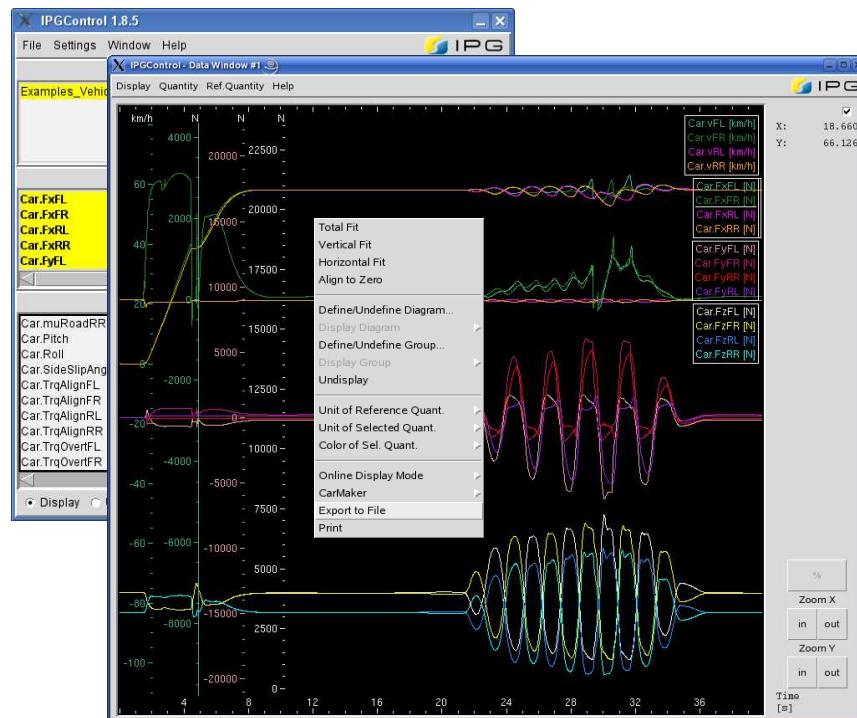


Figure 6.30: Calling the export functionality from the diagram

Before exporting the data, you may define your preferences for the export process:

<b>File Format</b>	A scrolling menu enables easy exportation of simulation data in one of the following formats:
	<ul style="list-style-type: none"> <li>• ASCII, separated by one of the following column delimiter (.csv): comma, semicolon, blank or tab</li> <li>• CSV format, e.g. a text format file with a column delimiter to be defined</li> <li>• XLS format, e.g. Microsoft Excel format</li> </ul>
<b>File Name / Location</b>	You can define the name and the location of the output file in a file browser.
<b>Quantity Unit</b>	You have to choose the units for the data to be exported. It can be either the units used in the diagram ( <i>Displayed unit</i> ) or the SI units ( <i>Original (SI) unit</i> ).

**Column Heading** You can choose whether the units will be displayed in the column heading or not.

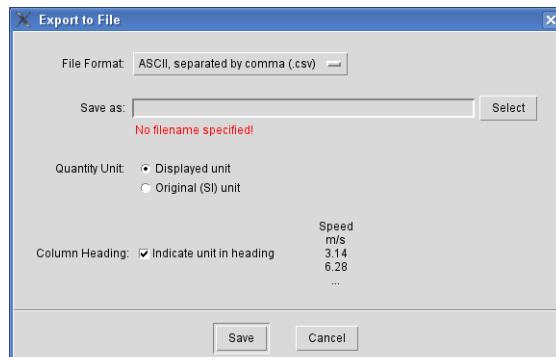


Figure 6.31: Export To File dialog window



Please note that the data set needs to be reconnected with the currently running CarMaker application once the export is finished. Otherwise IPGControl will no longer display any graphs of the running simulation.

To reconnect the data set, go to the IPGControl data window, right-click on the data set and select *Reconnect/Reload*.

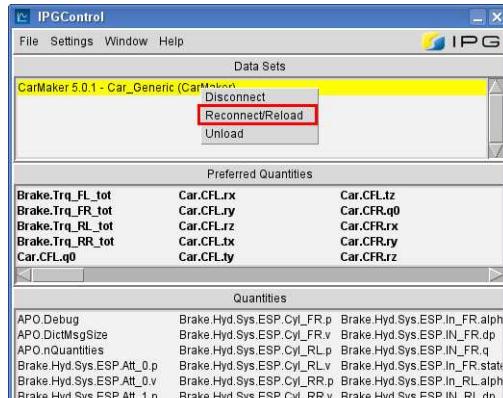


Figure 6.32: Reconnecting the data set with the currently running CarMaker application

## 6.6.6 Converting Tables with resutil

If you want to convert the whole result file to another format instead of only some parameters you can use the utility called *resutil*. Using this tool, \*.erg files can be converted to fortran binary or ASCII format and vice versa.

**Example** `resutil -a -om ascii -o ABC_good Brake_105823.erg`

In this example, an ASCII file named ABC\_good.dat is generated using the binary results file from the TestRun named "Brake". With additional options the file can be reduced to the desired size. Unwanted quantities can be filtered, the time domain can be limited and/or shifted and the resolution can be reduced by enlarging the minimal time step, as in the following example:

**Example** `resutil -s Car.ax,Car.ay,I0.ABSActive,Brake.TrFL -start 10 -end 15 -dtmin 0.01 -toff -10 -om ascii -o ABC_good Brake_105823.erg`

More detailed help can be found by entering the *resutil* command at the command prompt without specifying any options.

## 6.7 CarMaker Archive Service

In addition to the storage of results, you also have the possibility to save all parameter files that are required to reproduce these results. It means that an archive will be created at the end of each simulation including all files which were required by CarMaker to build the simulation results, e.g. TestRun, Vehicle and SimParameter files. The list of files contained in this archive is freely configurable.

- Activation** To activate the CarMaker Archive Service, select *Settings > Archive TestRun* on the CarMaker main GUI.

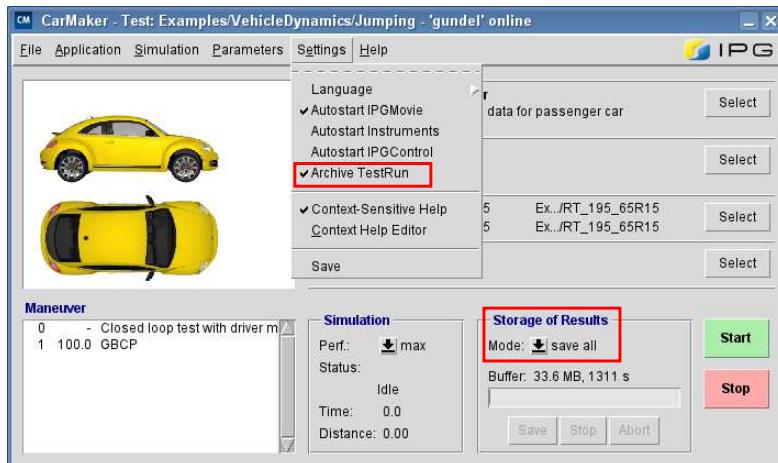


Figure 6.33: Activating the archiving functionality

Furthermore, the *Storage of Results* mode in the lower right corner of the CarMaker main GUI must be configured as *save all*. To configure the CarMaker Archive Service, select *Application > Archive Service* on the CarMaker GUI. The following window will appear.

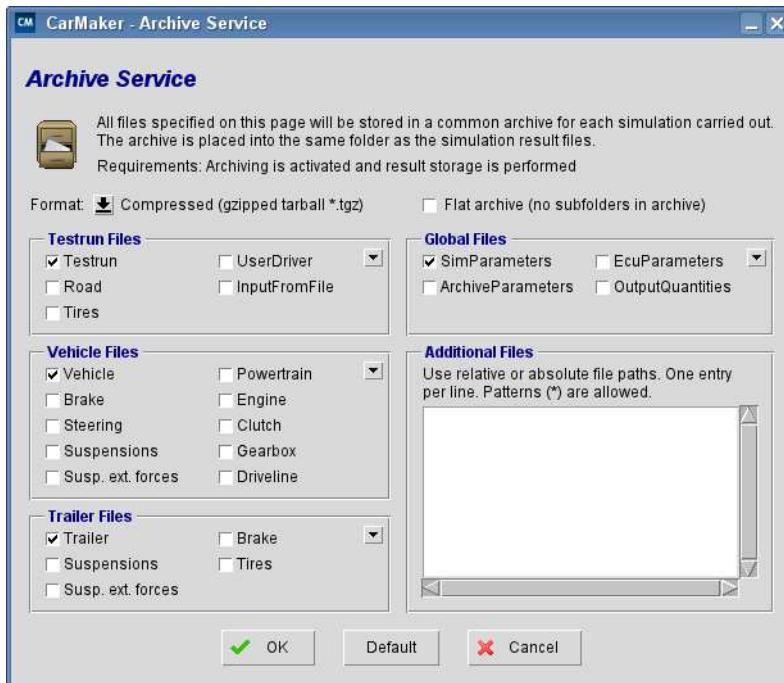


Figure 6.34: CarMaker - Archive Service

<b>Format</b>	At this point, you can select the file format of the resulting TestRun archive. Available formats are *.tgz, *.zip and uncompressed. Using Microsoft Windows, the default format is *.zip, on all other operating systems the default format is *.tgz.
<b>Flat Archive</b>	If activated, the resulting TestRun archive will consist of a single directory containing all archived files. If deactivated, the resulting TestRun archive will obtain the directory structure of the files to be archived. Per default, this option is deactivated.
<b>Testrun/ Vehicle/ Trailer/ Global Files</b>	By activating the tick boxes you can select the files which you would like to archive. By holding down one of the black arrows, you can select or deselect all files at once. The setup will be saved automatically for the next time. By clicking on the <i>Default</i> button, the default settings can be restored.
<b>Additional Files</b>	List of additional files to be archived after the completed TestRun. The following rules apply: <ul style="list-style-type: none"><li>• Paths may be absolute or relative to the project directory</li><li>• One entry per line</li><li>• Entries with wild cards are allowed, e.g. the entry <i>src/*</i> archives all files in the <i>src</i> directory</li></ul>

## Chapter 7

# Animation with IPGMovie

IPGMovie is a very powerful visualization tool that offers a lot of possibilities. Read more about it in this chapter!

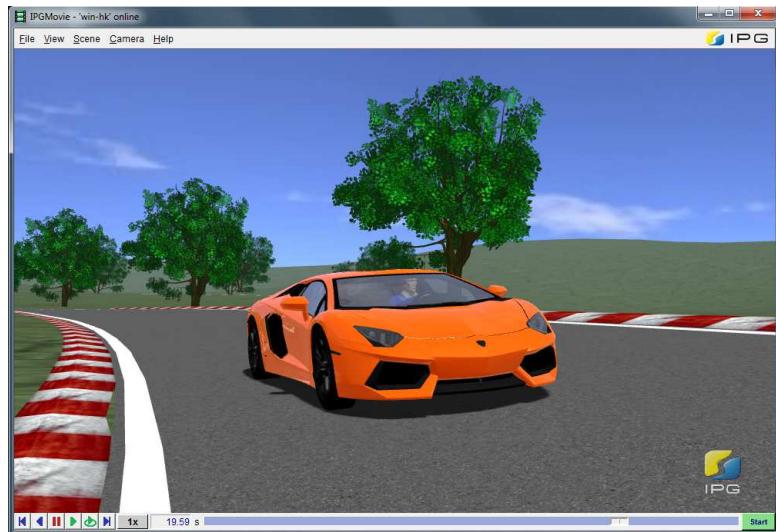


Figure 7.1: A racing circuit in IPGMovie



IPGMovie offers the functionality of an online animation. This means that the current simulation data is provided without delay - the virtual world is directly pictured during the simulation. By loading external result files you also have the possibility to watch the animation of a TestRun which has been carried out before. This is called the offline animation.



If you are using a computer with two graphics devices installed and IPGMovie is not running on the most performant one, you will be informed about this by an information dialog at the start of IPGMovie with the recommendation to select the right device.

## 7.1 User Interface Basics

### 7.1.1 Navigating with the camera

- Zooming** Zooming in and out in IPGMovie can be performed by scrolling the wheel of the mouse or by moving the mouse while pressing the Shift key and the left mouse button at the same time.
- Moving** To change the point of view you can keep the left mouse button pressed and move the mouse up and down or left and right.
- Camera** To manipulate the view direction you have another possibility: in the top menu of the IPGMovie window select *Camera*. At this point, you will find six pre-configured views: bird's eye view, left side, right side, sitting in vehicle (A), sitting in vehicle (B), default. If you defined a sensor on your car, another option will appear automatically with the name of the sensor, showing its perspective. Additionally, camera views are defined by default for the first five traffic objects within your TestRun. Above this, you can also define your own view under *Camera > Settings*.

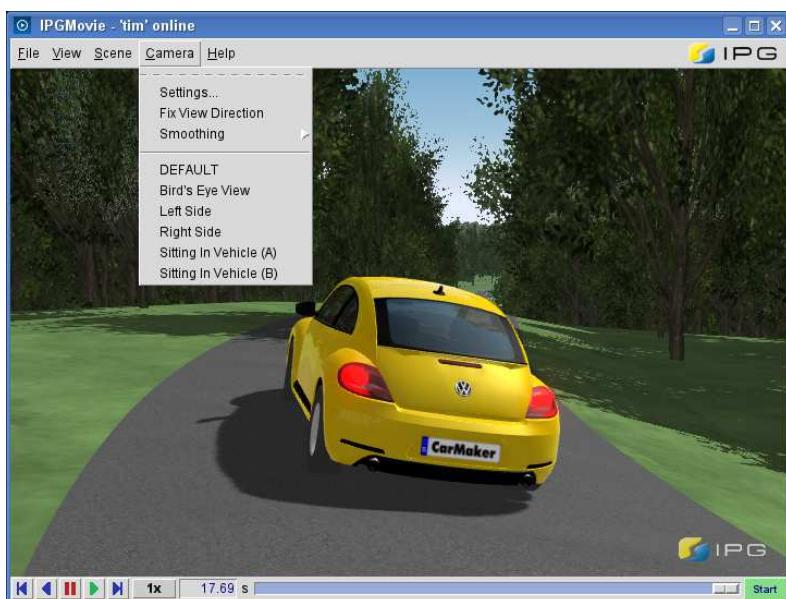


Figure 7.2: Selecting predefined camera views

If *Camera > Fix View Direction* is checked, the camera does not look into the same directory as the car, but look into a fixed direction instead. To see the difference try turning this on and off during a TestRun on a road with many curves.

*Camera > Smoothing* describes how soft the camera moves with the vehicle. When *off* is selected, the camera is mounted on an invisible arm which has a fixed length. Another possibility is an arm with a variable length, which leads to a delay in the reaction on the movements of the car like braking or accelerating. By selecting *low*, *medium* or *high*, you choose the intensity of the length variation.

### 7.1.2 Display options

- The View menu** In the menu *View* you can find settings related to the appearance of the animation and the animation window.

For information about *New Window* and *Split Window*, have a look at [section 7.2.8 'Multiple Views with IPGMovie' on page 415](#).

**Size** lets you choose between several predefined animation window sizes and a full screen mode without window frame and footer. Alternatively, the mouse can also be used to resize the window.



The smaller the window, the fewer pixels have to be rendered and the faster the vehicle simulation can be performed.

The Menu entry *Video Data Stream* belongs to an add-on feature of CarMaker which is not available in all editions of CarMaker. This menu entry activates / deactivates the data stream. To activate the display, you have to select it in the Overlay Menu. For more information about this, please have a look at [section 7.7 'Video Data Stream' on page 450](#).

Within the menus *Overlay - Top Left* and *Overlay - Top Right* you have the possibility to add the velocity, the rotational speed, the steering wheel angle, a detailed sensor view, a g-g diagram, a hybrid powertrain overview, a top view of the road and traffic or the configured Video Data Stream to the upper left and right side of the IPGMovie window. If Video Data Stream views are defined (see [section 7.7 'Video Data Stream' on page 450](#)), the previews will be available here as well.

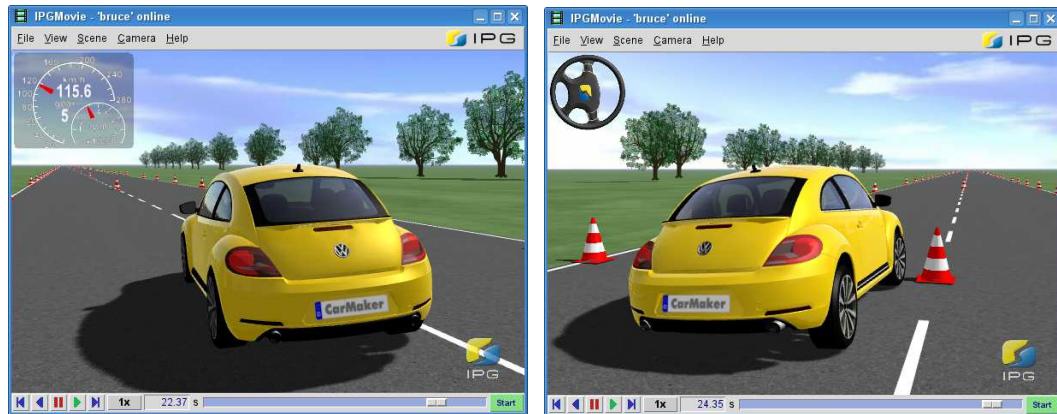


Figure 7.3: Velocity, rotational speed and steering wheel angle in IPGMovie

In Road Preview mode the overlays that are specific to the simulation will not be shown

Table 7.1: overlay shown according to the current display mode

Overlay	Shown in simulation mode	Shown in preview mode
Speed, Speed & RPM, Speed & Powertrain	X	
Road view	X	X
GG diagram	X	
Sensor view	X	
Steering wheel	X	
VDS preview	X	



Figure 7.4: Road view, g-g diagram and detailed sensor view in IPGMovie

The menu entry *Show* allows you to fine control several displaying options

- *Show > ABRAXAS*: this option activates the ABRAXAS view. You will find more information in the [section 'ABRAXAS' on page 402](#).
- *Show > Forces*: using this option, the tire forces are displayed at each wheel. The longitudinal and lateral forces as well as the wheel loads are shown as colored bars.
- *Show > Mass Balls*: shows the coupling masses in the ABRAXAS mode.
- *Show > Sensors*: If the sensors are configured in the vehicle data set, then they can be displayed in IPGMovie. The field of view of the sensor is depicted by a colored beam and can be seen in IPGMovie
- *Show > Line Detection*: if a line detection sensor is activated in the vehicle data set and this option is checked, the function of the sensor will be displayed in IPGMovie.
- *Show > Lights*: with this option, the display of braking lights, foglights and indicators is enabled.  
Warning: these are visual state change indicators, not actual lights. To get actual lights, please refer to [section 7.5 'Active Lights' on page 432](#).
- *Show > Command Markers*: you can place minimaneuver commands at fixed points on the road (see [section 'DrivMan Cmd' on page 64](#)). To indicate a road section with action command is passed, yellow diamond-shaped command markers can be used.
- *Show > Traffic Objects*: enables the display of one or more traffic objects which are defined with IPGTraffic (see [section 4.8 'Traffic' on page 127](#)).
- *Show > Driver*: only available in MotorcycleMaker
- *Show > Axles*: controls the display of the axles of multi-body suspension systems.
- *Show > Pitch & Roll Axles*: with this option, the pitch & roll axles are displayed.

*Quality Settings* provides a dialog with rendering settings to fine tune the visual quality in accordance to your hardware capabilities. See [section 'Motion Data' on page 407](#) for more information.

*Further Settings* provides a dialog with advanced settings intended only for experienced users. To get more information about this, have a look at [section 7.3.1 'Further Settings' on page 421](#).

**Scene** In *Scene* menu, you can configure the scenery in IPGMovie. The following options are available:

- *Sun*: sets the position of the global light (sun). You will find more information [section 'Sun - Settings' on page 402](#).

- *Background:* various background scenes are available for selection. Please find further information about on how to create a customized background in [section 7.3.4 'Displaying a User Defined Environment' on page 426](#).
- *Fog Mode:* With increasing distance the color of scene objects is blurred into *Fog*, yielding the impression of fog in the background.

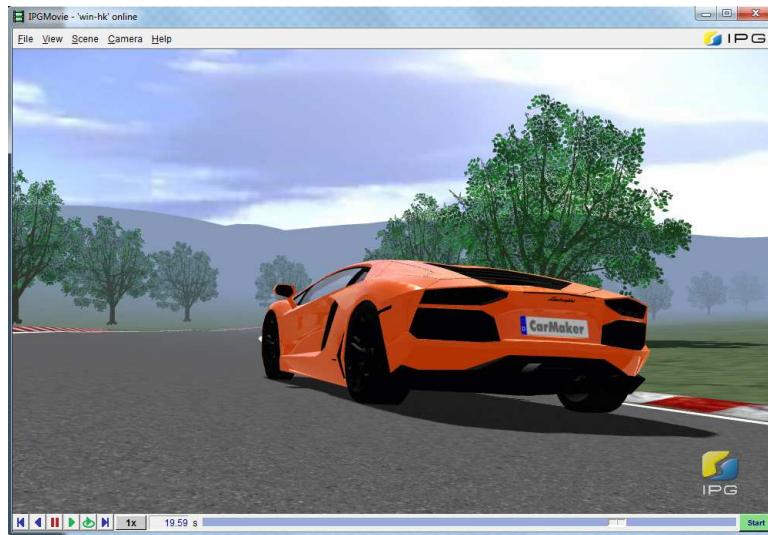


Figure 7.5: The visualization of fog in IPGMovie

- The option *Primary and Reference Vehicle* will be explained in [section 7.2.3 'Comparing two simulations' on page 405](#).



Figure 7.6: The Scene menu in IPGMovie

**ABRAXAS** Pressing the right mouse button in the animation window opens a context menu. With the first option *ABRAXAS* you can activate the ABRAXAS view. Now, your vehicle is no longer displayed as an object file but is stripped down to a basic geometry. It shows you the outer edges of the car, its tires and the moment of the steering wheel, the same applies for the optional trailer.

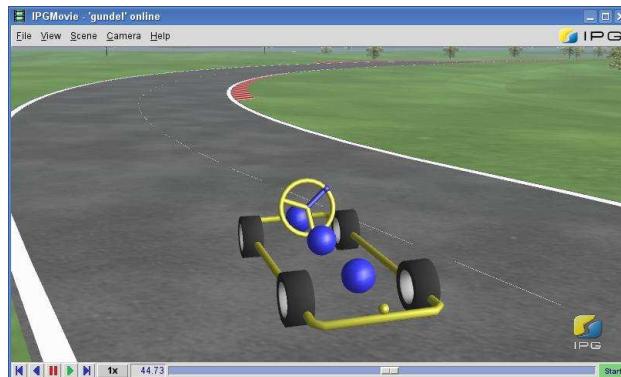


Figure 7.7: Abraxas view

The blue balls in the ABRAXAS model symbolize the coupling masses of the vehicle. Three masses in total (one on each of the two axles and one in the vehicle's center of gravity) are calculated out of the vehicle's overall mass and the moment of inertia in y-direction. The coupling mass is a quantity that describes the transfer from one axle to the other in case of an oscillation excitation. More information on the calculation of these masses can be found in respective literature about oscillation.

The yellow ball at the rear symbolizes the hitch point where a trailer can be coupled.

The ABRAXAS can also be activated in the IPGMovie menu under *View > Show > ABRAXAS*.



If you would like to use the ABRAXAS mode either for the vehicle or for the trailer and not for both, you can deselect the movie geometry file under *Parameters > Car/Trailer > Misc. > Movie Geometry* on the CarMaker main GUI.

#### Sun - Settings

Using this dialog, you have three different possibilities to change the position of the sun. The settings of this dialogue will be saved in the *Movie* folder of the current project directory in a file called *Environment.cfg*.

*Attached to the Camera* is the default setting when the visual quality *Performance* is selected. It offers a light source that is positioned at the active cameras location (a small offset exists) and can be used as some kind of flashlight. This mode is ideal for inspecting some objects details, e.g. the axles of the ego vehicle, because no parts will be darkened by shadows.

Using the option *Fixed Position - Definition With Angles*, it is possible to define the cardinal direction (azimuth) and height (elevation) angles via the slide controls or by typing in the degrees.

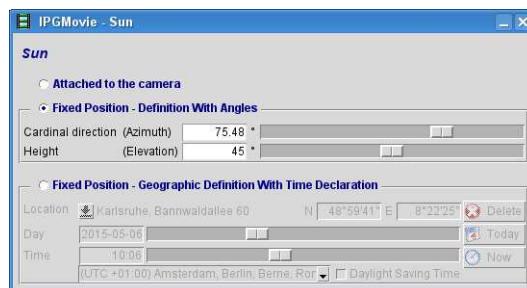


Figure 7.8: Sun - Fixed Position - Definition With Angles

Using the option *Fixed Position - Geographic Definition With Time Declaration*, you can define the position of the sun according to a specific location on earth. You can use one of the preset locations or add another location. The mapping of location names and its specific settings (longitude, latitude and utc-offset) will be saved in the *Movie* folder of the current project directory in a file called *LocationPresets.cfg*

To add a new location, select *add new location* in the location dropdown menu. After you named the new location, you must fill in the longitude and latitude as well as selecting the correct utc-offset from the dropdown menu. When this is done, you can adjust the day and time using the respective sliders. Furthermore, select the right time zone via the dropdown menu at the left bottom. If required, you can also activate the daylight saving time by selecting the tick box *Daylight Saving Time*.

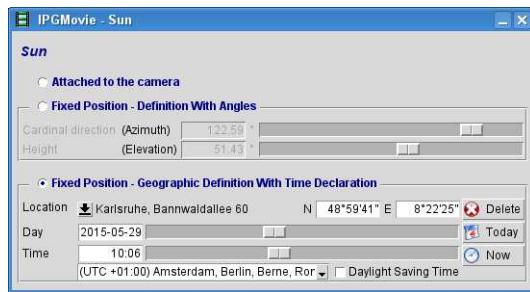


Figure 7.9: Sun - Fixed Position - Geographic Definition With Time Declaration

## 7.2 Advanced features

### 7.2.1 Simulation control with IPGMovie

The green start button at the lower right corner is linked to the start button in the CarMaker GUI, you do not have to switch to the CarMaker GUI for (re-)running the simulation.



Figure 7.10: Simulation control with IPGMovie

### 7.2.2 Replay

In the offline mode, you can view a simulation again after the TestRun has finished. If you want to take a closer look at a certain scene, you do not need to run the whole simulation again. With the buttons and slider at the footer of the IPGMovie window, you can rewind the

animation, play it again, stop it at any point or move the slider to the position of interest. With a double-click on the simulation time, an exact time can be entered and IPGMovie will jump to the corresponding scene.

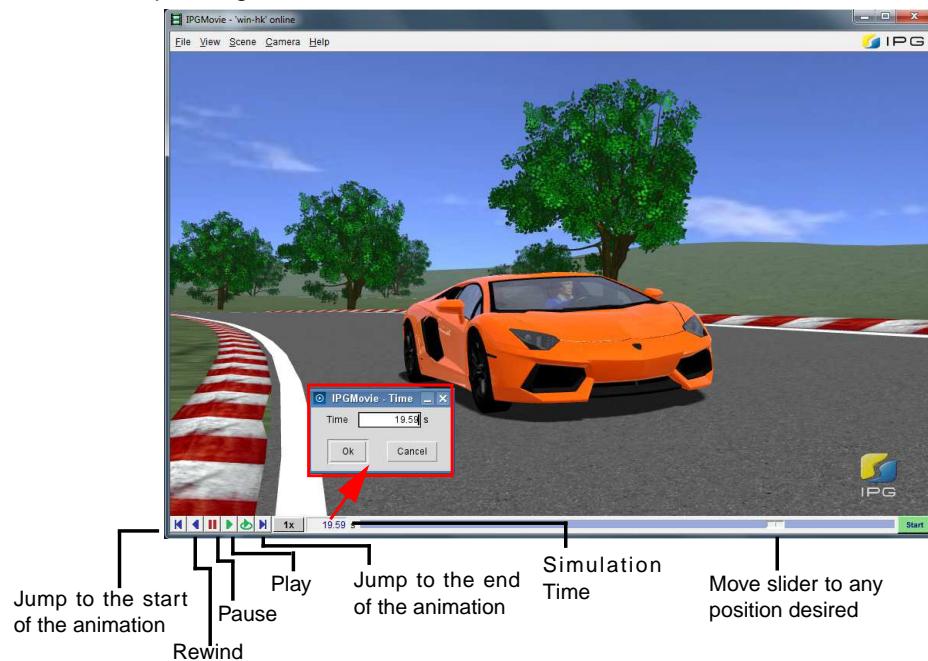


Figure 7.11: Footer options in IPGMovie

During the replay, you can also vary the speed of the animation. However, in the online mode of IPGMovie (when the simulation is running in the background) the animation speed will follow the simulation performance settings in the CarMaker GUI (realtime, faster or slower than realtime).

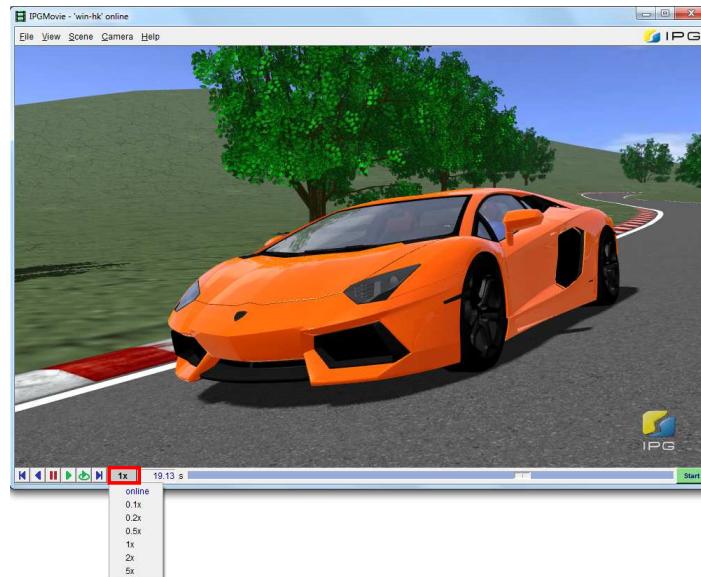


Figure 7.12: Changing the animation speed in IPGMovie

Keyboard shortcuts exist as well for each footer functionality. Additionally, the animation can be repeated by moving through every frame. The shortcuts are listed in the Table 7.2 (RTfac: realtime factor, default=1):

Table 7.2: IPGMovie shortcuts

Key	Action	Description
space	stop/play toggle	play start/stop
b	backward	reverse play, decrementing time
Left/Right	step	Time step $\pm 0.04 \times RTfac$
Shift- Left/Right	big step	Time step $\pm 1.00 \times RTfac$
Control- Left/Right	small step	Time step $\pm 0.01 \times RTfac$

Using the IPGMovie replay mode, the Instruments panel and the IPGControl diagram windows are automatically connected, in case the two applications have already been used during the simulation. A vertical position marker indicates the current position in the IPG-Control diagram.



Figure 7.13: Replay in IPGMovie with synchronized replay in the IPGControl diagram window

### 7.2.3 Comparing two simulations

You have the possibility to display two different vehicles at the same time in IPGMovie. This is a very easy way to observe the effects of the changes you made in the vehicle configuration. The reference vehicle does not even need to be the same as your primary vehicle – you can compare completely different configurations from different TestRuns.

For this, you need to activate the option *Primary and Reference Vehicle* in the menu *Scene*. With this, you ensure that the display of two vehicles at the same time is possible.

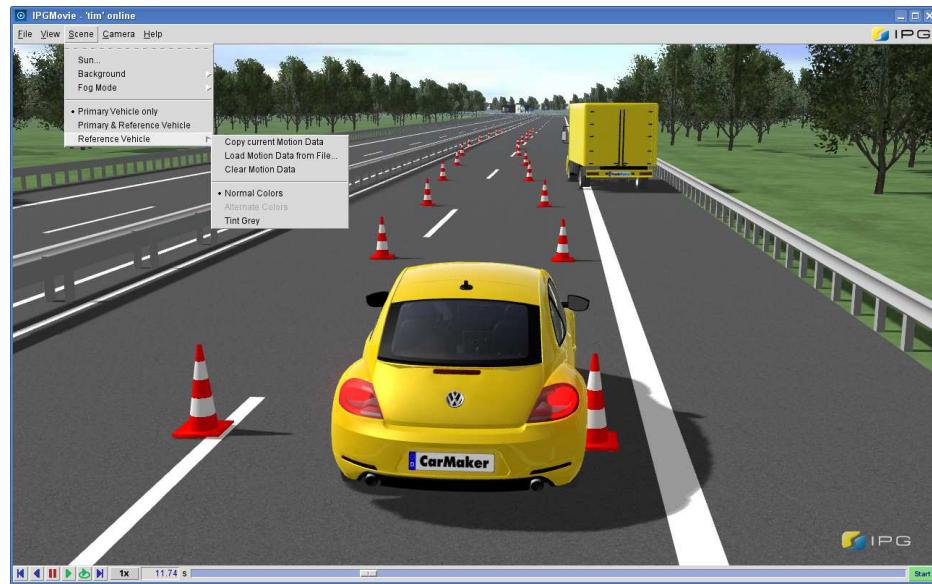


Figure 7.14: Loading data for the reference vehicle

The second step would be to define the data source for the reference vehicle. There are two options available:

- *Copy current Motion Data*: the data of the last simulation performed is taken as reference. However, it will always remain the same: if you perform multiple simulations, there will be no automatic update to copy the the last simulation as reference. The reference data stays the same as long as you do not choose another data set.
- *Load Motion Data from File*: the reference vehicle can also be from another TestRun. For this, a simulation of this TestRun must have been performed before and the data saved to a file.

By default, the reference vehicle is displayed in a tinted color tone for a better distinction. It can be changed to normal colors in the same menu.

The example below shows a primary and a reference vehicle which vary in the driver settings. The driver at the reference vehicle is allowed to have a corner cutting coefficient of 0.95 whereas the same value was set 0.2 for the driver of the primary vehicle (see [section 4.7.2 'User parameterized Driver' on page 118](#)).

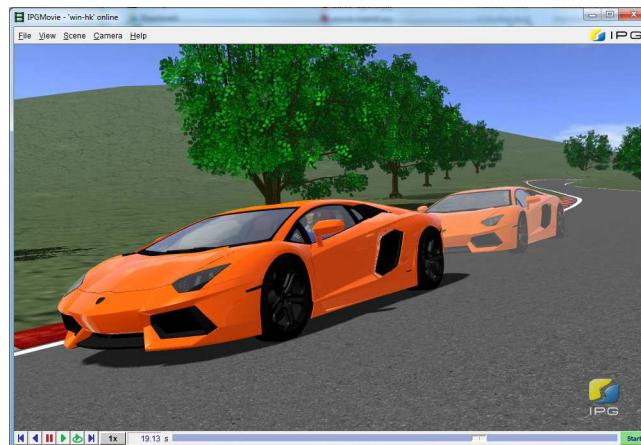


Figure 7.15: Comparison of primary and reference vehicle with different driver settings

## 7.2.4 Loading simulation data

### Motion Data

Via the menu *File > Load > Motion Data...* you can load the data vectors of a simulation, provided that

- you have saved the results (see [section 6.5.1 'Starting the Saving Process' on page 382](#)) in an ERG-file
- you have selected the appropriate quantities (see [section 6.5.2 'Configuring the Saving Process' on page 383](#))

This option will only load the motion data of vehicle through the saved quantities in the result file. Neither the road nor the 3D movie geometry of the objects declared in the TestRun will be used.

### Simulation Data

If you wish to replay a simulation with all its visual components, you can do this with the menu item *File > Load > Simulation Data...*

Choosing an ERG-file will lead to IPGMovie investigating more sources of information in order to recreate the simulation as it was when it was saved.

The conditions are the same as for motion data, with one more:

- all external files related to the visualization (3D models, digitalized roads, etc.) used in the simulation must be available, either in the product folder, in the project folder, or in any other DataPool defined in the user interface of CarMaker

### Missing Quantities

In case IPGMovie finds out from the test run information that some quantities are expected but not found in the ERG-file (for instance: you have defined some traffic objects, but their related quantities are not selected to be saved), you will get a warning message, informing you that the list of the missing quantities is stored in the file *<project folder>/MISSING\_QUANTITIES.txt*. Please open this file to get more information about which quantities may need to be selected when saving the simulation results.

Missing quantities will not prevent the offline simulation to be replayed, but some elements of the simulation will not be animated at all (for instance: the traffic objects which quantities are missing, will not move).



Hint: Use the predefined storage setting "OutputQuantities\_Anim" under *Application > Output Quantities > Configuration > Load* to make sure all signals required for the replay are saved in the result file.

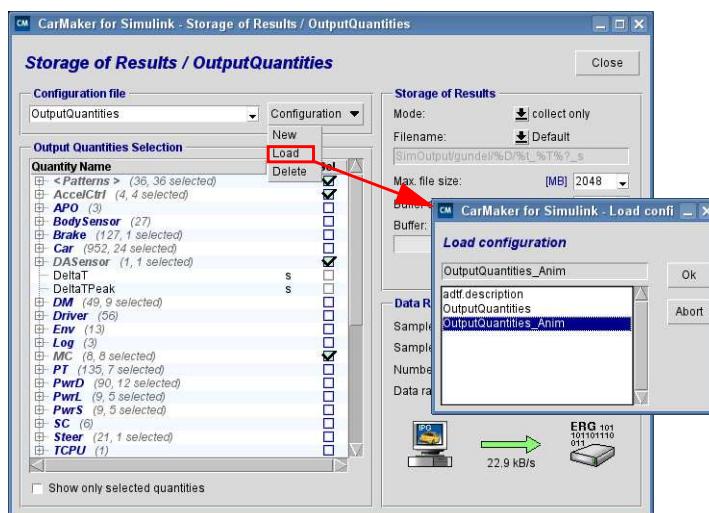


Figure 7.16: Predefined storage configuration for the a replay of the result file

## 7.2.5 Quality Settings

Via the menu *View > Quality Settings*, it is possible to select among three predefined quality settings: *Performance*, *Balanced* and *Quality*.

Table 7.3: Predefined Visual Quality Settings

Quality Presets	Shadows	Active Lights	Puddles	Blooming	Reflections	Pylons
Performance	Static	No	Static	Off	Normal	Low Quality
Balanced	Dynamic	Yes	Dynamic	Off	High	High Quality
Quality	Dynamic	Yes	Dynamic	On	Dynamic	High Quality

Additionally, you can adapt the settings by yourself. You can activate and define the intensity of shadows, motion blur, puddles, blooming and (dynamic) reflections for the simulation. The puddles have a static reflection in the normal quality setting. Using the *Balanced* and *Quality* settings, the shadows and reflections are dynamic with a better image quality.

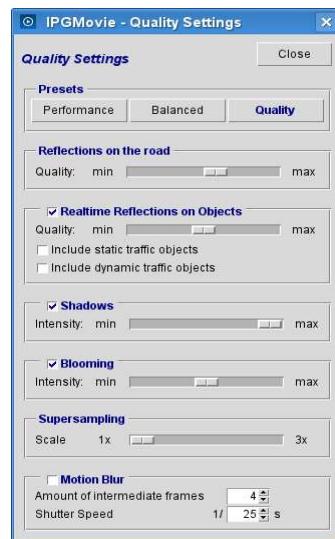


Figure 7.17: Rendering Quality of the Visual Effects

**Shadows and reflections** When the high quality shadows are not selected, standard quality shadows are used. They are aligned to the defined dimensions of the vehicle. If the dimensions are not set correctly, the shadows will also have the wrong size. The dimensions can either be defined in the Vehicle Data Set Editor or in the Traffic dialog.

This also applies for the dynamic reflections on vehicles and objects. They are enabled automatically if the highest quality setting is selected, otherwise you can activate them manually. In this dialog, you can additionally adjust the quality of the dynamic reflections and activate the display of static and dynamic traffic objects within the dynamic reflections.



The *dynamic shadows and reflections* and especially the display of static and dynamic traffic objects within the reflections can increase the rendering time significantly. If you notice any performance issues, try to deactivate the dynamic shadows and/or reflections or to reduce their quality.

The display of *Reflections on the road* (as seen with puddles) as well as the *Blooming Effect* can be activated or deactivated via the respective tick boxes.



For more information regarding puddles, have a look at the section '[Road Painting](#)' on page [86](#).

The reflections on objects and vehicles can be activated by selecting the respective tick box. To add reflectivity to a specific material, add the line  $Km\ r\ g\ b\ lod$  to the respective material within the material file.  $r\ g\ b$  is the amount of reflected environment over the Red, Green and Blue channels. A value of 0.0 means the base color is 100% visible, the reflections 0%, whereas a value of 1.0 means the base color is 0% visible, the reflections 100%.

$lod$  stands for the level of detail of the reflection. This helps simulate the roughness of the material. A value of 0 is the level with no loss of details, used for perfectly smooth surfaces like mirrors. A value of 1 is the level with the highest loss of details. Use it for very rough surfaces like brushed metal.

The following table shows you some examples, which you can use as a guideline:

Table 7.4: Guidelines for specific materials

Material	km
Glass	0.5 0.5 0.5 0.2
Chrome	0.75 0.75 0.75 0.15
Mirror	1.0 1.0 1.0 0.1
Painting (normal)	0.125 0.125 0.125 0.6
Painting (matt)	0.125 0.125 0.125 0.8
Painting (polished)	0.125 0.125 0.125 0.4

**Motion blur** *Motion blur* is available, if you have the *VDS license extension*.

When the *Motion blur* effect is enabled via the Rendering Quality dialog, you can finetune the accuracy of the effect. Therefore, you can define the amount of intermediate frames (Default: 4) as well as the shutter speed of the camera (Default: 1/25s).



Be aware that the motion blur effect can increase the rendering time significantly. Reducing the amount of intermediate frames and increasing the shutter speed can have a positive effect on performance. Furthermore it is possible to deactivate the motion blur effect in IPG-Movie and to activate it for the Video Export and/or the Video Data Stream (VDS) separately. To get more information regarding the Video Export, have a look at [section 7.2.9 'Exporting videos and images' on page 417](#).



To activate the motion blur effect in the VDS, you have to set two new keys in the VDS.cfg file. The key *VDS.MBlur.nSamp* controls the amount of intermediary images taken while the shutter is open. A value of 0 means that the motion blur effect is turned off. To turn it on, you need to set the value to 2 at minimum. The second value is *VDS.MBlur.ShSpeed*, which controls the shutter speed.

**Supersampling** *Supersampling* is available at all quality presets if you have the *VDS license extension*.

*Supersampling* is a post processing filtering that reduces the amount of jagged edges that may appear and affects the whole picture. It renders the picture in a higher resolution than needed for displaying and then downsamples this picture with a bilinear filtering. The scale of the picture used for rendering can be set by the user. It can take the values 1x (no supersampling) or 2x (height and width of the rendered picture are multiplied by 2).



Figure 7.18: Showing different supersampling factors: 1x (left), 1.5x (center), 2x (right)



With supersampling set to 2x the whole picture will be antialiased, which means

- the final picture will look slightly blurrier than without supersampling
- more graphics memory and rendering time is required



Which means that higher values may not necessarily improve the visual quality, while demanding unnecessary graphics resources. We therefore recommend to test values between 1.25 and 2.0 until you meet an acceptable trade-off: improved picture quality versus performance loss.

## 7.2.6 Advanced Camera Settings

IPGMovie offers a special dialog to fine tune the camera view. The advanced camera settings are saved project-specific in the *Camera.cfg* file in the Movie subfolder of the respective project directory. The Camera Settings dialog now has a context menu which allows to save and reload the camera settings.

To open this dialog, select *Camera > Settings...* in the menu bar of IPGMovie.

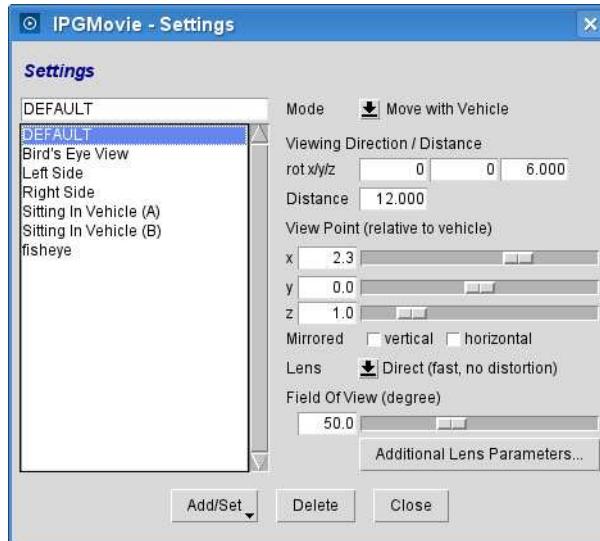


Figure 7.19: Adding a new view in the Camera Settings dialog

A new view can be added by defining a name in the entry field on the left side and clicking **Add/Set**. The new view is now directly selectable in the *Camera* menu. The camera settings can be adapted by changing the *Viewing Directing* and *Distance*, the position of the *View Point* and the *Field of View*.

Under *Mode* you can define if the camera should follow the vehicle from outside or if the camera is fixed on the vehicle body. In case a vehicle with flexible body is used, you can either mount the camera on Body A or B.

The checkboxes *Mirrored vertical / horizontal* allow you to flip the picture, horizontally, vertically or both. This can be useful when a specific view is used e.g. viewing a rear mirror.

Further information on the different lens types available and their special configurations can be found in [section 7.2.7 'Camera lens types' on page 411](#).

### Additional camera rotation

It is possible to rotate the camera around the axes of the vehicle coordinate system Fr1 (see Reference Manual for further information on axis systems).

To parameterize the viewing direction, the order of the rotations is important. To facilitate the calculation of the requested Euler angles, a specific tool can be called from the window context menu (right-click in the window).

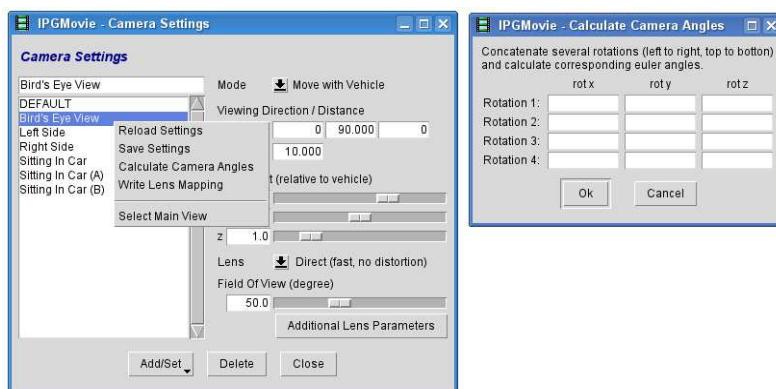


Figure 7.20: IPGMovie - Camera Angles Calculator

This tool allows to define up to 4 consecutive rotations and automatically calculates the resulting Euler angles needed in the *Camera Settings* window.

## 7.2.7 Camera lens types

### Rectilinear projection

The rectilinear projection model is a pinhole camera model, i.e. a camera with a lens that acts like a single small aperture. Light from a scene passes straight through this single point and projects an inverted image on the camera sensor. The final image is obtained by re-inverting the camera sensor image.

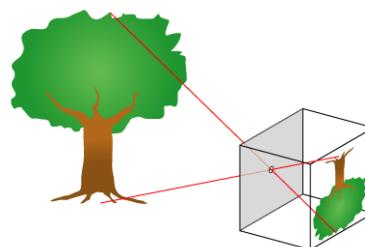


Figure 7.21: Principle of a Pinhole Camera

### Fisheye lenses

With circular fisheye lenses, the image is circular and generally inscribed in the camera sensor area, therefore the black area in the frame corners. The fisheye lenses can be classified according to their mapping function, i.e. the way the image is distorted.

#### Equidistant, linear

The equidistant linear fisheye lens transforms the object angle of view linearly into a distance with respect to the image center point.



Figure 7.22: Fisheye, equidistant, linear

#### Equal area

This model applies for lenses that use an equal-area or equi-solid-angle projection as a mapping function. It means that equal solid angles in three-dimensional space project into equal areas in the image plane.



Figure 7.23: Fisheye, equal area

#### Orthographic

The mapping function for this lens model is an orthographic projection which is a form of parallel projection. All the projection lines are orthogonal to the projection plane resulting in every plane of the scene appearing in affine transformation on the viewing surface. The result of the orthographic projection would be the view of a sphere, with the view mapped to the outside, seen from a far distance.



Figure 7.24: Fisheye, orthographic

**Stereographic** This model uses a stereographic projection which is a mapping that projects a sphere onto a plane. The projection is defined on the entire sphere, except at one point (projection point). This mapping preserves angles but neither the distances nor the area of objects.



Figure 7.25: Fisheye, stereographic

### Fisheye lens settings

Additional parameters can be specified in the appropriate window.

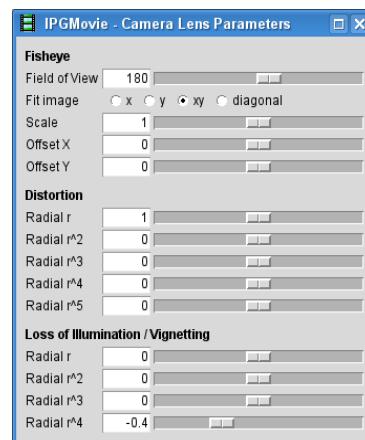


Figure 7.26: Camera Lens Parameters

**WARNING:** Many of those parameters are enabled only if the optional module VDS is available.

#### Field of View

The camera field of view can be specified in the range of 90 to 250 degrees.

#### Fit Image mode

Several image fit modes for fisheye images are available:

- x: the fisheye circle is adapted to fill the window in the horizontal direction. This mode has to be used to simulate the cameras mounted on the vehicle.
- y: the fisheye circle is adapted to fit the vertical dimension.
- xy: the full fisheye circle is displayed and fits the smaller dimension (horizontal or vertical).
- diagonal: the fisheye circle is adapted so that the diameter fits the length of the window diagonal. It is a full frame mode: no black areas are displayed in the frame.



Figure 7.27: Fisheye image fit mode

**Scaling / Offsets**

- Scale: this parameter is a scaling factor for the fisheye image.
- Offset X: this parameter enables to shift the image in the horizontal direction.
- Offset Y: this parameter enables to shift the image in the vertical direction.

**Distortion**

In addition to the mapping function of the fisheye lens, a distortion can be defined by a polynomial of degree 5 whose coefficients are adjustable:

$$Distance(r) = Scale \times (a \cdot r + b \cdot r^2 + c \cdot r^3 + d \cdot r^4 + e \cdot r^5) \quad (\text{EQ } 2)$$

where *Distance* = newly calculated radius for the considered pixel

*r* = radius calculated by the mapping function of the fisheye lens

*a, b, c, d, e* = user defined coefficients in the graphical interface

The coefficients *a,b,c,d* and *e* respectively correspond to Radial *r*, Radial *r<sup>2</sup>*, Radial *r<sup>3</sup>*, Radial *r<sup>4</sup>* and Radial *r<sup>5</sup>*.

**Loss of  
Illumination /  
Vignetting**

The vignetting is an unintended effect that consists of a reduction of the brightness or saturation at the image periphery compared to the image center. It is caused by the angle-dependence of the camera sensor. Light incident on the sensor at a right angle produces a stronger signal than light hitting it at an oblique angle.

This effect can be modeled by a polynomial of degree 4 which coefficients are adjustable:

$$Intensity(r) = 1 + a \cdot r + b \cdot r^2 + c \cdot r^3 + d \cdot r^4 \quad (\text{EQ } 3)$$

where *Intensity* = light intensity for the considered pixel

*r* = distance of the pixel from the image center

*a, b, c, d* = user defined coefficients in the graphical interface

The coefficients *a,b,c* and *d* respectively correspond to Radial *r*, Radial *r<sup>2</sup>*, Radial *r<sup>3</sup>*, Radial *r<sup>4</sup>*.

Original image



Image with vignetting effect



Figure 7.28: Vignetting effect

- Grid** By clicking on the parameter GUI using the right mouse button, you can select a grid. According to the camera field view, IPGMovie needs to generate 4 or 5 direct images before generating a fisheye image. When the Grid parameter is not set to 0, it is possible to visualize how the different images have been assembled. This parameter specifies the grid step.

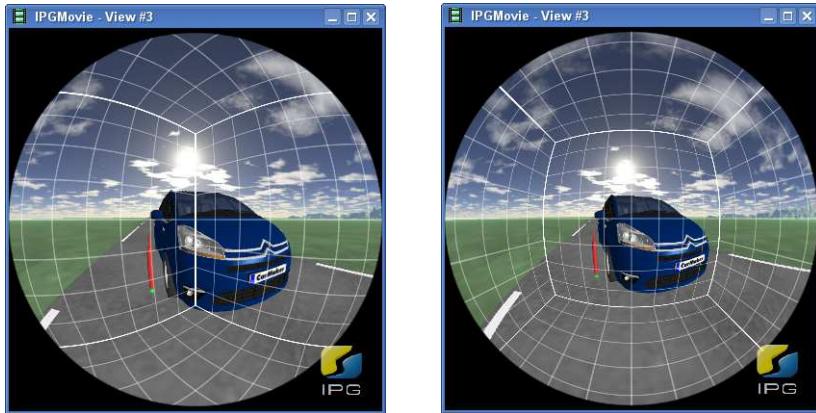


Figure 7.29: Generation of fisheye images

### 7.2.8 Multiple Views with IPGMovie

IPGMovie is able to display up to 4 fully synchronized, independently configurable view windows of the animation. Each additional view window can be opened from the menu *View* in the IPGMovie main window or from the right-click context menu in each window.

Additionally, each single window can be split in up to 4 independently configurable views. Obviously, more views and/or more windows demand more performance from your system. You can select four different types of gaps between the views: none, tiny, medium and big.

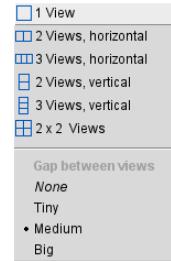


Figure 7.30: Split View Options

- New Window** By clicking on *New Window* in the menu *View*, a new window is displayed with another view of the animation. By default, the newly created view has the same camera settings as the main view, but new camera settings can be applied.

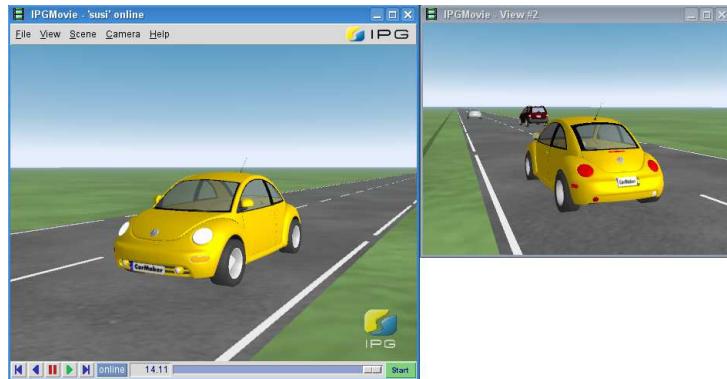


Figure 7.31: Adding a View to IPGMovie

The additional view is still linked with the main view; a displacement of the slider in the main view will affect the additional view as well. The IPG logo displayed in the bottom right hand corner indicates which view is active at the moment. Modifying the camera settings will only affect the active view.

**Split Window** The button *Split Window* opens up a menu which enables you to split the currently activated view into 2, 3 or 4 views in the horizontal or vertical direction as well as a 2x2 view. To close split views and return to a single one, select *1 View*. All view windows can be split.

**Camera Settings** Each of the additional views can have different camera settings. The settings can be modified interactively (when the view is active), saved and displayed again by default.

Once the camera view settings have been defined for a given view, the configuration can be saved simply by specifying a name for the configuration and by clicking on the button *Add/Set*. Then it is possible to apply the same settings to another view by activating the view and clicking on the configuration name in the list of available configurations.

It is also possible to parameterize the default settings for a given view by pushing the button *Add/Set* for a moment. A list with different possible choices to define default settings for all existing views will appear.

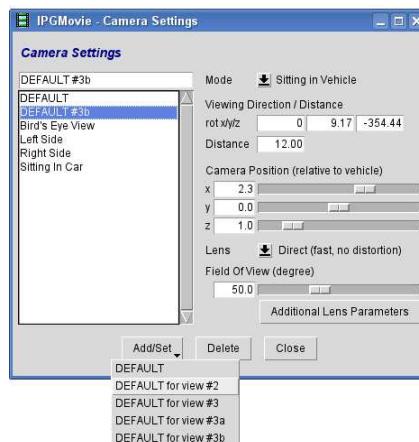


Figure 7.32: Camera Settings in IPGMovie

In this list, you will find entries for each of the existing windows:

- **DEFAULT:** The chosen configuration will be applied to the IPGMovie main window.
- **DEFAULT for view #X:Y:** Camera settings for the view Y in window X. The index starts at 0, therefore, for instance, the default camera settings for the second view of the first window would be saved under the name *DEFAULT for View#0:1*

## Running IPGMovie on a Different Host

IPGMovie can also be started on a separate computer and connected to the running application via the network. This even works across different architectures (MS Windows, Linux and XENO). Prerequisite is that you have (remote) access to the CarMaker project directory (via NFS or SMB share) and that the project directory path is valid for all hosts.

IPGMovie needs to be started manually with the option *apphost* using the command line as described below:

Start the command shell and move to the */hil/version/GUI/* folder of your CarMaker installation directory using *cd*:

```
cd \IPG\hil\version\GUI
```

Then, call the IPGMovie executable with the option *-apphost Hostname*, where *Hostname* is the name of the target host the CarMaker application runs at:

```
Movie.exe -apphost gundel
```

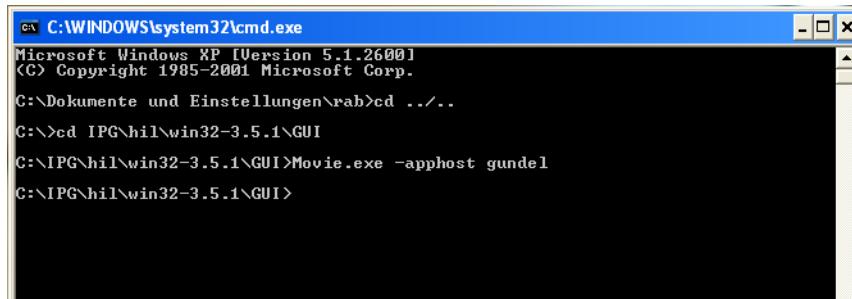


Figure 7.33: Starting IPGMovie manually and connect it to an application running on a different host

For each computer you want to use IPGMovie, an own license file is required.



### 7.2.9 Exporting videos and images

IPGMovie allows to export images or even whole videos, which can then be embedded into e.g. presentations. Select *File > Export* from the main menu bar.

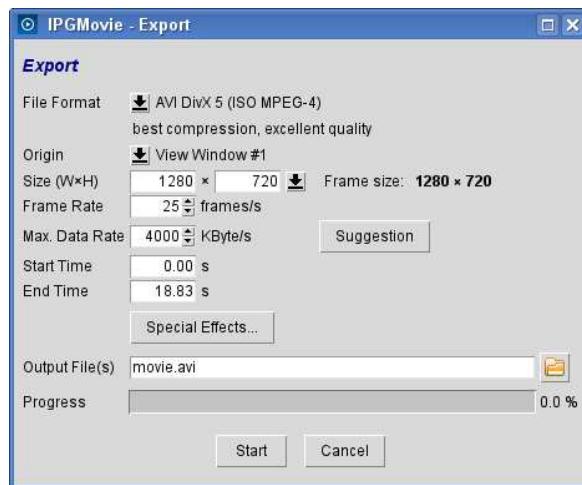


Figure 7.34: Export options in IPGMovie

Depending on what you intend to export (image or video), you can choose different file formats. The list of choices for *File Format* partly depends on the video codecs currently installed on your system.

Different views in a split window will be exported together, including the gap between the views. The default value of *View Size* is probably not the same as the current size of your IPGMovie view and it should be kept small to avoid too big files. Any frame size can be specified, but it is recommended that you stick to one of the predefined common sizes. *Frame Size* indicates the actual size of the frame, including all views of the window and the gap between them.

*Frame Rate* and *Maximum Data Rate* can also be defined, as well as a *Start* and *End Time* of the recording. The higher the data rate, the higher the resulting video quality, but the bigger the resulting video file will be.

<b>Rendering time</b>	Some factors will increase the time required to generate the video data. The more views in a view window and the bigger the view is, the longer the time needed to render the video.
-----------------------	--

The simulation of fisheye lenses will also increase the video rendering time. This will be even more noticeable, if the frame size of the video / image to be exported differs from the size of the view shown on screen.

Therefore it is recommended, to use the same size for the export view and for the display view when rendering fisheye lenses simulations.

Activating the Video Data Stream will also increase the video rendering time.

**Rendering size** For videos, the size of each view will be reduced to the nearest multiple of 4. For instance, if you give a view size of 497x349 pixels, then the generated video will have a size of 496x348 pixels. If you have selected a margin between the views, this rule applies for the margin, too.

Reducing the size of each view makes sure, that all views always have the same size and the compression artefacts at the border between two views are reduced.

For instance: if you have 2 views with the size of 350x400, the video will have the size 700x400. While 700 is a multiple of 4, 350 is not, resulting in possible compression artefacts where the 2 views coincide.

**Missing FBO support** To render Video Data Stream and video/images to be exported, IPGMovie uses an off-screen rendering technique called "Framebuffer Objects", which needs to be supported by your graphic hardware. If this is not the case, you will be warned (message box) that the on screen display of IPGMovie will be used instead of rendering the animation/image. Please make sure, that the window being exported is not covered at all with any other window.

**Video Codecs** On Windows systems, you will need to install 3rd party software that will take care of encoding the frames from IPGMovie to video files. Such a software is called a *codec* (compression of the words COding and DECoding). We recommend

- XVid
  - MPEG4-compatible, great compression level
  - Download: <http://www.xvid.org>
  - IMPORTANT - during the installation process, when the option is offered, make sure you check "Vfw Interface"
- Lagarith Lossless Video codec
  - Lossless compression: low compression level
  - Download: <http://lags.leetcode.net/codec.html>

## 7.2.10 Connect to Google Earth

From IPGMovie a direct connection to Google Earth can be established. All you need is an installation of the Google Earth software - please find more information on supported versions in the Release Notes.

The *File* menu item *Connect Google Earth* automatically starts Google Earth and connects

it to IPGMovie. If GCS coordinates are specified in the Road parameterization (see [section 'Global Road Attributes' on page 43](#)), the vehicle position in Google Earth is updated once per second.

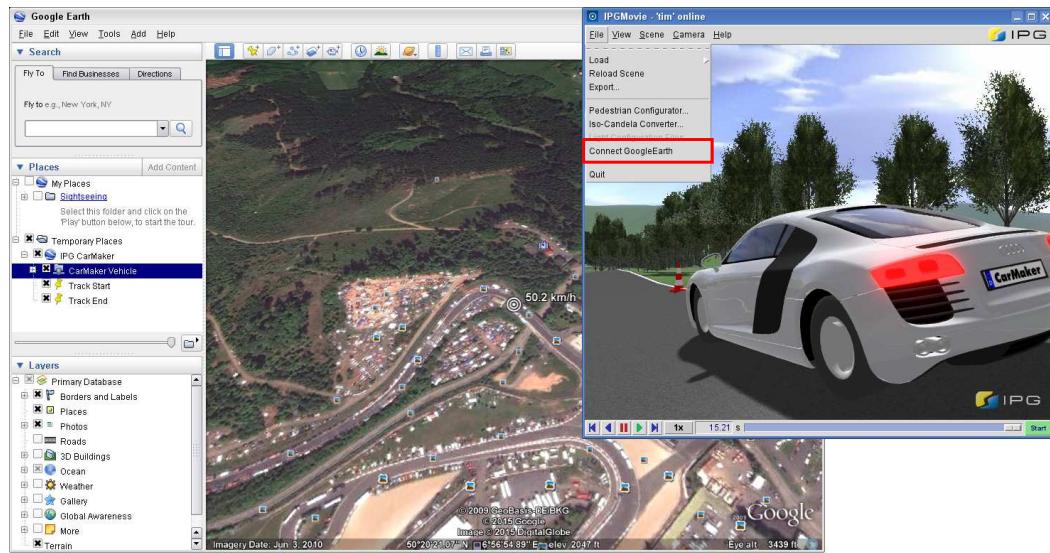


Figure 7.35: Connection of Google Earth with IPGMovie showing the Hockenheim TestRun



As an example take the TestRun Examples > CarMakerFunctions > IPGRoad > Hockenheim.

### 7.2.11 Integrating Dynamic Objects

IPGMovie offers the opportunity to integrate dynamic objects. You can use quantities calculated in your simulation to change the color, the direction, the size or the scaling of the inserted object depending on the movements of the car or any other calculated quantities during the TestRun.

In this example, the feature is demonstrated by using the included VhclBox.tclgeo file. In order to implement this, you have to copy the VhclBox.tclgeo file as well as the \*.obj and \*.mtl file of the desired car from the *Movie* directory of the installation path into the *Movie* directory of your current project directory. To link both files, you need to insert `#Include "VhclBox.tclgeo"` into the *IPG-MOVIE-INFO* section of the \*.obj file of the car.



As an example take the TestRun Examples > VehicleDynamics > Braking.



Figure 7.36: Colored box on top of vehicle during braking maneuver

## 7.2.12 Integrating Light Assistance Systems

IPGMovie offers the possibility to visualize light assistance systems like dynamic cornering light in your simulation. At this point we will demonstrate this feature based on the included *VhclLights.tclgeo* file.

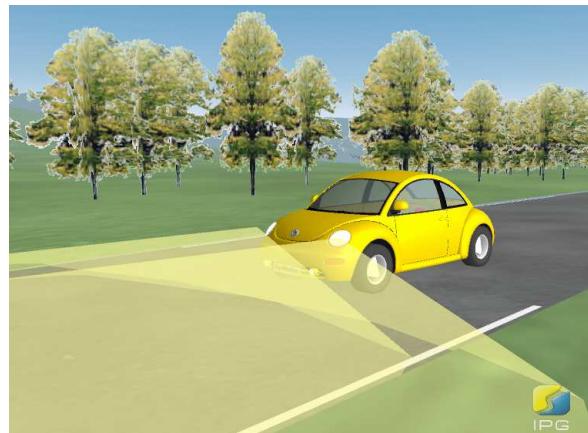


Figure 7.37: Visualized lights in IPGMovie

To integrate this light assistance system, you need to copy the *VhclLights.tclgeo* file as well as the \*.obj and \*.mtl files of the respective car out of the *Movie* subfolder of your installation directory into the *Movie* subfolder of your project directory. In order to link the files, you have to insert `#Include VhclLights.tclgeo` into the *IPG-MOVIE-INFO* section of the \*.obj file.

```

1 #
2 # VW New Beetle
3 # Converted by the DEEP Exploration 2.00.1211
4 # Right Hemisphere, LTD
5 # http://www.righthemisphere.com/
6 #
7 # Adapted for IPG-MOVIE 3.2
8 # 2005-6-8, IPG Automotive GmbH, www.ipg.de
9 #
10 ### BEGIN IPG-MOVIE-INFO
11 # NumPlate PRODUCT 4.05 0.00 0.29 0 180 0.56 0.11
12 # NumPlate PRODUCT 0.08 0.00 0.55 12 0 0.35 0.13
13 # Translate 1.98 0 0.2
14 # Scale 0.305 0.305 0.305
15 # Rotate 90 0 0.1
16 # Include VhclLights.tclgeo
17 ### END IPG-MOVIE-INFO
18
19 mtllib VW_NewBeetle_2005.mtl
20

```

Figure 7.38: Registering the VehicleLights.tclgeo extension in the vehicle graphic file

If the required quantities are not provided by your simulation environment, you can add the following MiniManeuver Commands into the Minimaneuver Commands field of the maneuver definition window (see [section 4.5.8 'Minimaneuver Command Language' on page 107](#)):

```

Eval Qu::BeamL.rx=0
Eval Qu::BeamL.ry=0
Eval Qu::BeamL.rz=0
Eval Qu::BeamR.rx=0
Eval Qu::BeamR.ry=0
Eval Qu::BeamR.rz=0

```

By setting *set ObjBeamL* and *set ObjBeamR* into the *VhclLights.tclgeo* file, you can generate a permanently defined shape of the lights on the ground.

## 7.3 Fine-tuning the animation

### 7.3.1 Further Settings

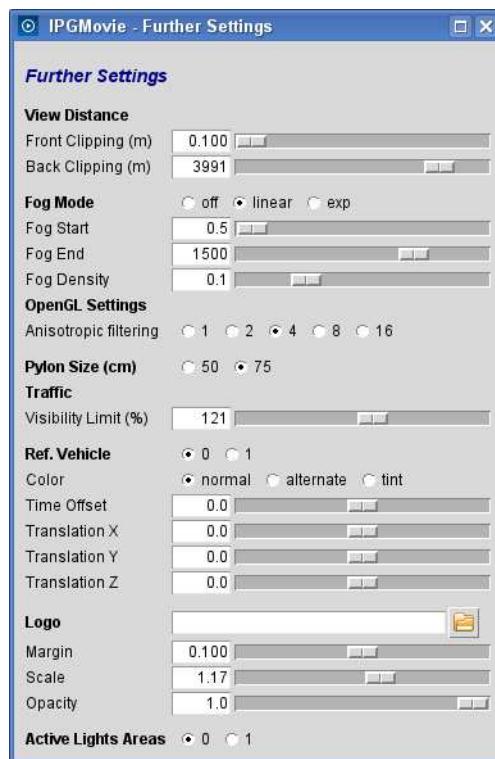


Figure 7.39: The expert parameters dialog

**Front/Back Clipping** The front and back clipping values define the maximum view distance (back) and minimum view distance (front) in meters. Reducing the overall view distance may drastically improve the performance.

**Fog Mode** You can choose between three different fog modes for the visualization: *off*, *linear* and *exp*. Using the mode *linear*, it is possible to define the front and back distance in meters to the camera using the sliders *Fog Start* and *Fog End*. In this case, the *density* slider has no effect. Using the option *exp*, which offers optically different styles of fog, only the *Fog Density* slider is active. If you need more information regarding this topic, please have a look at the OpenGL appendix.



Please note, that the settings of the Fog Mode will be reset when another skybox is selected for the visualization.

**Traffic** In the *Traffic* section of the *Expert Parameters*, you can set the visibility of the traffic objects using the slider *Visibility Limit (%)*. If the traffic objects are located beyond this specified limitation, their bounding boxes are displayed instead of the 3D object which improves the performance.

**OpenGL Settings** At this point, you can activate *Anisotropic Filtering*. This option improves the texture quality and sharpness. If you need more information regarding this topic, please have a look at the OpenGL appendix.

**Ref. Vehicle** If you have a reference vehicle activated in your simulation, you have the possibility to modify its appearance or to disable it at this point (0=off, 1=on). The option *Color* empowers you to change the visualization of the car. Setting a value for *Time Offset* allows you to let the reference vehicle start earlier or later than the ego vehicle. By modifying *Trans X/Y/Z*, the reference vehicle can be moved to the desired position.



Please note, that the settings of the Reference Vehicle will be reset on the next start of the simulation.

### Logo

The IPG Automotive logo is displayed at the bottom right corner of the IPGMovie window. An additional user defined logo can be placed in the opposite corner with this option.

At this point, the path to the logo can be selected; supported image file formats are PNG, GIF and JPEG. The dialog also allows to set the distance to the window border, scaling (view size) and transparency of the image.

- For a transparent background either the PNG (preferred, most flexible) or GIF format should be used. The JPEG format is not suited for this purpose.
- The size of the logo should be 64x64, 128x128 or 256x256 pixels. Bigger textures are not guaranteed to work. Depending on the OpenGL implementation of your graphics hardware vendor there might also be a restriction for the texture's size to be a power of two.

You can find an example logo "My Logo" in your CarMaker installation directory under *IPG/hil/version/GUI/Textures/myLogo.png*.

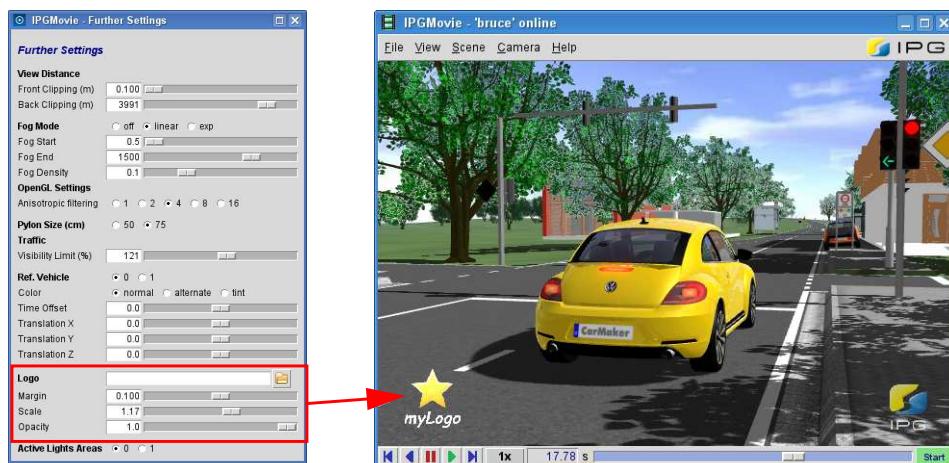


Figure 7.40: Inserting your own logo in the animation

**Active Light Areas** To get more information about Active Lights, please have a look at [section 7.5 'Active Lights' on page 432](#).

### 7.3.2 The IPGMovie Interface in IPGRoad

In the road dialog window in the CarMaker GUI (*Parameters > Road*) you can define a road in terms of segments, bumps, markers, vehicle start position etc., whose information are completely related to the vehicle simulation. A realistically looking animation, however, requires some additional aspects to be taken into account, which is the purpose of the *IPGMovie Interface* tab in the road dialog.

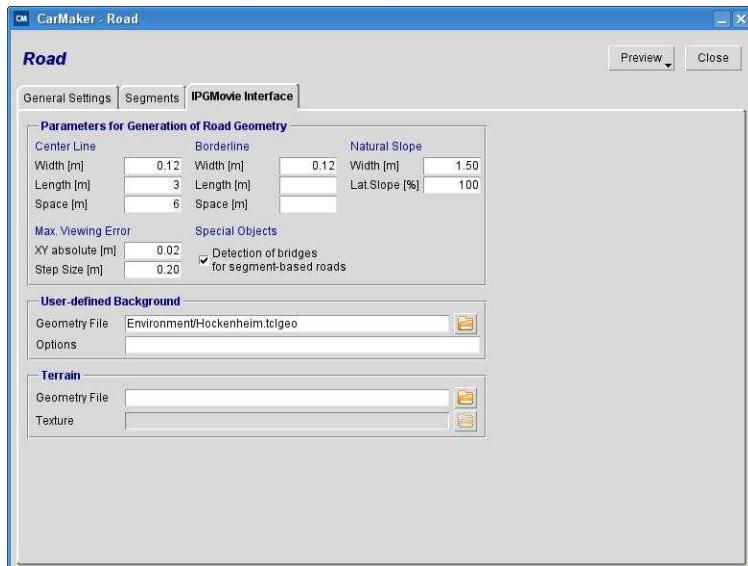


Figure 7.41: IPGMovie Interface in IPGRoad

**Center Line** Here you can define the layout of the center line. You can set the width of the white stripe in the middle of the road, its length and a spacing. If you would like to have a continuous line, set spacing to 0. If you do not need a center line at all just set all parameters to zero.

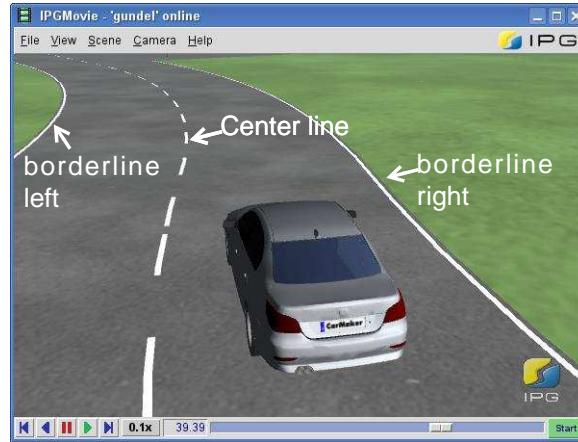


Figure 7.42: The dashed center line and continuous borderlines

**Borderline** Defines the layout of the lines that limit the road on both sides. The parameters are the same as for *Center Line*: width, length and spacing of the line.

**Natural Slope** Here you can define the slope of the environment left and right to the road.

**Max. Viewing Error** This parameter describes the maximum allowed deviation of displayed surfaces from the actual geometry, as this affects the visual quality of the displayed scene. Smaller values mean finer granulation (e.g. smoother curves) at the cost of an increased number of polygons that need to be displayed for the scene. Useful values are in the range of 0.01 m to 0.05 m.

**User-Defined Background** IPGMovie offers the possibility to change the environment in the background. Predefined files are included in the Movie/Environment folder of the CarMaker installation directory. However, you can also create your own environment using \*.tclgeo files (script language Tcl/Tk) or object files. These script-based environment files allow the integration of road signs, trees, buildings and many other items for beautification. To see examples, please go to the *Movie/Environment* subfolder of your installation directory. Any \*.tclgeo file can be opened with a text editor.

Saving the files to the *Movie/Environment* folder of your project directory makes them available in this box. Be aware, that your object files do not contain any textures as they cannot be loaded into IPGMovie so far.



As an example take the TestRun Examples > CarMakerFunctions > IPGRoad > Road\_Roundabout.

### 7.3.3 Using Alternative Textures for Tree Strips

Since CarMaker 5.1 tree strips can be defined with an Additional Parameter, which is used to change the texture used for the trees of the strip (see [section 'Tree Strip\(s\)'](#) for more detailed information about creating tree strips).

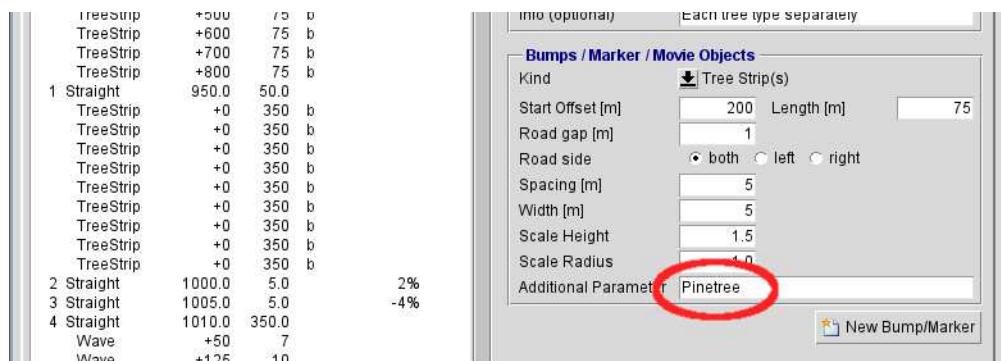


Figure 7.43: IPGRoad dialog: Additional Parameters for tree strips

The given name follows a simple convention: *<name>* points to the file *Movie/Textures/Trees/<name>.png*. For instance, the parameter *Pinetree* points to the file *Movie/Textures/Trees/Pinetree.png*.

Some examples are available in the installation path, but following this naming convention you can use your own textures, when properly stored in *<project>/Movie/Textures/Trees*.

If the texture file is not found, or if the field is left blank, the standard texture will be used.



Figure 7.44: on the left: the standard tree, on the right, an alternative

If you mix several tree strips with alternative textures, you can easily get a mixed tree forest.

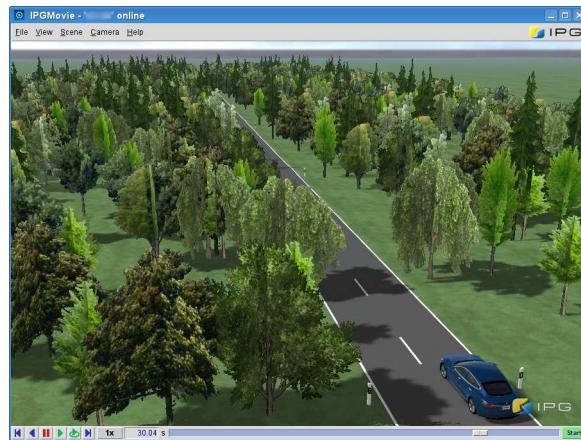


Figure 7.45: mixed tree forest



#### Tips and tricks for the alternative textures

- Do not use the exact same parameters for consecutive tree strips.
- Only the PNG file format is supported. It offers support for transparency and compression and is therefore the most suitable.
- The dimension of the picture should not exceed 512 pixels (1024, at most).
- If you get a white or black border on the tree, here is a picture processing method that will help
  - Open the PNG file in a picture editor that supports layers.
  - Set the transparency threshold to 50% (or 127, if the values go from 0 to 255). If some pixels appear that clearly do not belong to the tree (sky, background, etc.), remove them.
  - If needed, reduce the size of the picture so that its biggest dimension is 512 or 1024.
  - Duplicate the picture in 4 layers. You should now have a total of 5 layers.
  - Offset the first layer to -2 in X, making sure that the layer does not wrap around. Set its opacity to 1%.
  - Offset the 2nd layer to +2 in X, making sure that the layer does not wrap around. Set its opacity to 1%.
  - Same process for the 3rd and 4th layers, except that the offset will be of respectively +2 and -2 in Y
  - Do not modify the last (5th) layer.
  - If need be, merge all the layers.
  - Save. The black or white border should be gone.

### 7.3.4 Displaying a User Defined Environment

You may want to use your own environment in addition to the predefined skybox backgrounds you find in the *Scene > Background* menu of IPGMovie.

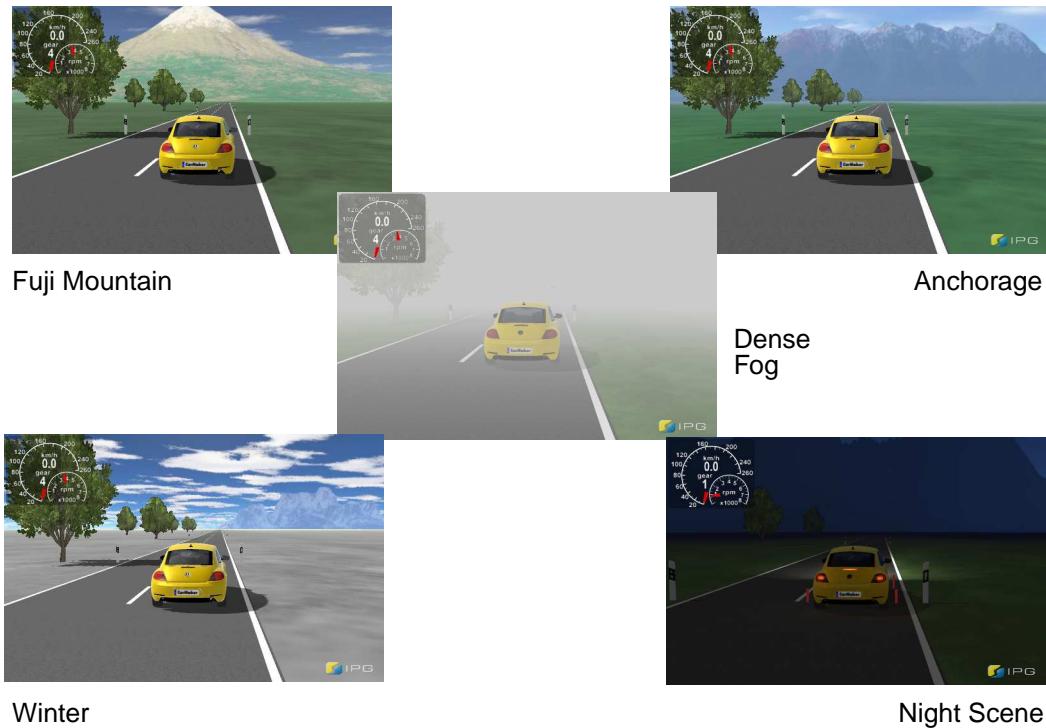


Figure 7.46: Different default backgrounds available in IPGMovie

To display your own background you need quadratic textures for the six sides of the skybox cube and a text file containing the background description.

- Texture files** The textures should have a resolution of 256x256, 512x512 (recommended) or 1024x1024 pixels. As file format \*.png or \*.jpeg should be chosen and a horizontal viewing angle of exactly 90 degrees is required.

```
files:  
Valley_%.jpg
```

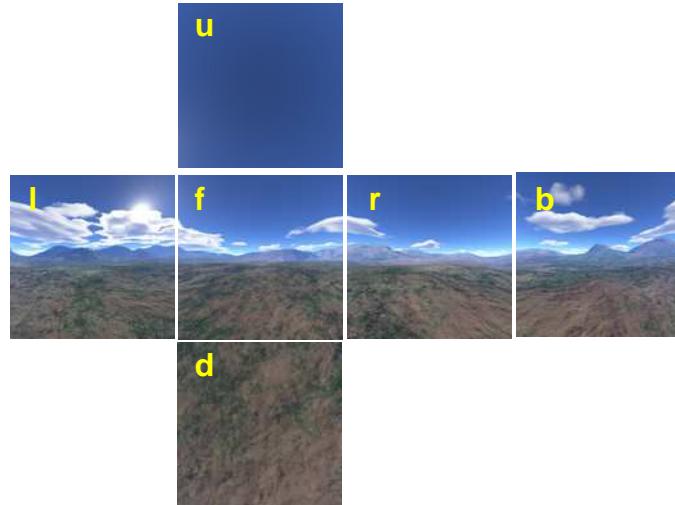


Figure 7.47: Layout and naming convention of the skybox cube

Each of the six textures that complete a skybox cube must have exactly the same resolution and must be named after the following naming convention:

```
name_%.ext
```

"name" is an arbitrary name, "ext" is the file extension (png, jpg) and "%" is the letter representing the side of the cube: f=front, b=back, l=left, r=right, u=up, d=down (see also [Figure 7.47](#)).

#### Description file

The background description file can be a simple text file. The example below defines a blank background by including six textures called *BlankBg\_%.png*. With the line *Name* you define what is displayed as the name of the skybox in the *Scene > Background* menu of IPGMovie.

```
Name  Blank Example
Files BlankBg_%.png
```

The description file should be placed in your CarMaker installation directory under *IPG/hil/version/GUI/Textures*, together with the six texture files. It should be called *name.skybox*. With the next restart of IPGMovie, the menu entries will be updated and the new skyboxes are added.

This directory also contains the files for the *BlankBg* example described above. To activate the example, simply rename file *BlankBg.example* to *BlankBg.skybox* and you will find an additional entry "Blank Example" in the *Scene > Background* menu after the next restart of IPGMovie.

The following additional attributes can be added optionally to the description file:

```
LightA r g b,
LightD r g b
LightS r g b
```

These attributes define the global directional light color (*LightA* = ambient, *LightD* = diffuse and *LightS* = specular). All values are in the range 0..1.

```
FogColor r g b
FogMode mode
FogDensity value
FogStart distance
FogEnd distance
```

At this point, the fog is parameterized. The fog color is given in rgb, according to the mode selected the fog can be turned off (mode: off) or spreads with linear (mode: lin) or two different exponential characteristics (mode: exp/exp2) between the start and end value. The fog density ranges from 0 to 0.4, the start and end distance specify the fog diffusion.

```
BgColor r g b alpha
```

It defines a blend color and a blend factor *alpha* (0..1) for the background skybox.

```
Ground r g b (alpha)
Road r g b (alpha)
```

These two parameters define the color of the ground level (default: green) and the road (default: grey).

```
Road-Spec specularity shininess
```

This key defines the specularity of the road material. The specularity is "the quantity used in 3D rendering which represents the amount of reflectivity a surface has. It is a key component in determining the brightness of specular highlights, along with shininess to determine the size of the highlights" (Wikipedia:Specularity).

The parameters must respect the following rules:

```
0.0 <= specularity <= 1.0
0.0 <= shininess <= 128.0
```

## 7.4 IPGMovie Object Files

### 7.4.1 General Information

The movie objects required to display your vehicle, trailer or traffic objects in IPGMovie can be generated by yourself. Common CAD software provides the option to export files in Wavefront \*.obj format. There are also various commercial websites offering object files for download.

Furthermore, IPGMovie provides the opportunity to integrate texture-files into your \*.obj-files.

For an optimal simulation performance, your movie object should not contain more than 50,000 facets (triangles). The scene generation may become too complex for the graphics display hardware, which is leading to low frame rates in IPGMovie.

#### Limits

If a 3D model contains more than 6,5 million vertices or more than 3,2 million faces, then IPGMovie will refuse to load it and a warning message will be shown.

#### Recommendations for Geometry Export

For the movie object export, we suggest you to follow these advises below:

- Generate the movie object in the Wavefront \*.obj format. You can also use the closed \*.mobj format. As alternatives, you can use the \*.dae (Collada) or \*.kmz (Sketchup Warehouse) formats. Other formats are not supported by IPGMovie.
- Do not generate splines or free forms, due to the fact that IPGMovie only understands triangles and polygons.
- During export do not forget to create a material file which contains the material data and associated color information.
- We also recommend to generate the normals for all points ("vn" coordinates in the \*.obj-file). This will enhance the shading of the object. If they are not defined, IPGMovie will try to generate them.

Once you have created your object file, it needs to be saved along with the material file in the *Movie* subfolder of your project directory; otherwise IPGMovie will not find it.

#### Integrating a geometry object

Some additional IPGMovie-specific information can be added to the object file. The information is contained in a comment block surrounded by two special lines "### BEGIN IPG-MOVIE-INFO" and "### END IPG-MOVIE-INFO". All lines in between have to start with a hash character (#) marking a comment line.

In human-readable ASCII based formats (\*.geo, \*.roadgeo, \*.obj and \*.tclobj files) the information can be found in a comment block at the beginning of the file.

In binary \*.3ds files, this information can be appended to the end of the file (e.g. with "cat info.txt >> geo.3ds"). If necessary, the information can be changed later using an editor capable of editing binary files.

**Example**

```
### BEGIN IPG-MOVIE-INFO
# Translate 2.05 0 0.61
# Scale 0.0258 0.0262 0.0256
# Rotate 90 0 0 1
# Rotate 90 1 0 0
# Include front.obj
### END IPG-MOVIE-INFO
```

This information is only evaluated by IPGMovie and will be ignored by other programs. In most cases, after editing the geometry data with another program the additional information will get lost. However, this information is crucial to make the geometry visible in IPGMovie.

The possible entries which may appear inside such a comment block will be explained below. Each of the directives may be specified multiple times. The transformations given by Scale, Translate and Rotate will be evaluated in the given order and determine the coordinate system for all geometry data contained in the file. They also affect all following NumPlate and Include statements.

Collada files also support some of those IPG-specific commands, but in this case the block is an XML block instead of such a `BEGIN ... END` block.

**Translate x y z** Translation of the geometry (offset) by x, y, z in [m].

**Example**

```
# Translate 2.05 0 0.61
```

**Scale sx sy sz** Scaling of the x/y/z axis by the factors sx, sy, sz.

**Example**

```
# Scale 0.0258 0.0262 0.0256
```

**Rotate alpha  
x y z** Rotation by an angle alpha around a vector x, y, z in [degree].

**Example**

```
# Rotate 90 0 0 1
```

**TwoSided** Modifies the behavior of the lightning model, so that the back sides of all facets are illuminated the same way as the front sides. This may help if the orientation of some parts of your geometry does not fit completely. Most often these parts appear too dark and without diffuse/specular lights. The preferred way to correct this is to switch the orientation of the concerned normals/facets. Use this option only if necessary.

**NumPlate name  
x y z ry rz wi he  
cur** Inserts a special geometry object - "number plate" at the defined position x, y, z.

The center of the rectangle sized wi (width) by he (height) will be positioned at x, y, z. Rotation around the y/z-axes is specified in degrees by ry, rz. The expression rz = ry = 0 denotes a number plate in the y-z-plane, whose front side can be seen when looking in x-direction. The last attribute cur defines the curvature of the NumPlate to adjust it to the geometry of the vehicle graphic file. Positive values bend the plate backwards, the default value is zero which means no bending.

You can either call up a predefined or a user-defined numplate which can be formatted as \*.jpg or \*.png. To use the predefined numplate, you have to call up *NumPlate PRODUCT*, to use the user-defined numplate you have to call up *NumPlate <Name>* or just <Name>. In the last case, the prefix will be added automatically. The numplates can be located in the *Movie* directory or its *Textures* subfolder. Otherwise, you can load the respective numplate via <Directory>\<Name>.

Example of a rear number plate:

**Example** # NumPlate PRODUCT 0.08 0.00 0.55 12 0 0.55 0.12

**Wheel.pos** Inserts a wheel to the geometry object (only relevant for Traffic Objects). The wheel is defined by its position *f1* = front left, *fr* = front right, *rl* = rear left, *rr* = rear right. The fifth item defines the radius *rad*, followed by the width *wi* and the rim diameter *rim* (0 means not specified, use default). Optionally, an external wheel geometry file can be loaded by stating the file name of the object file.

**Example**

```
# Wheel.fl 3.48 0.75 0.30 0.31 0.15 0 filename.obj
# Wheel.fr 3.48 -.75 0.30 0.31 0.15 0 filename.obj
# Wheel.rl 0.80 0.75 0.30 0.31 0.15 0 filename.obj
# Wheel.rr 0.80 -.75 0.30 0.31 0.15 0 filename.obj
```

**Include fname** In addition to the geometry information contained in the current file, a geometry file *fname* will be read. The format of *fname* is allowed to be different from that of the current file (but, of course, must be supported by IPGMovie).

**Example** # Include front.obj

Relative filenames are interpreted relative to the directory of the file containing the include statement.

#### Subscribe



If required, additional non-standard quantities which IPGMovie should take into account for the visualization can be added with the help of this option.

Please note that the update rate for subscribed quantities is 100Hz.

#### CullFace

This option offers you the possibility to reduce the amount of drawn polygons. If a closed body is visible, only the polygons of the outside area will be calculated. According to that, about the half of the polygons will not be calculated from IPGMovie which may improve your simulation performance. The outside area of the movie object is defined by the vectors of the polygon points.

#### Lights

IPGMovie provides the possibility to insert different types of lights into the simulation. To insert a light, it is required to copy the following line into the IPG MOVIE-INFO block inside the \*.obj/\*.mobj file of the respective car.

```
# Light <type> <x> <y> <z> <ry> <rz> <w> <h> <rounding: 0..1>
```

The variable *<type>* is the key to choose the type of the light you would like to insert into your simulation. The following types of lights can be simulated by IPGMovie:

- *Brake*: brake light
- *IndicatorL*: indicator light on the left side of the car
- *IndicatorR*: indicator light on the right side of the car
- *FogFrontL*: fog light on the front left side of the car
- *FogFrontR*: fog light on the front right side of the car
- *FogRear*: rear fog light
- *Reverse*: reversing light

The values named *<x>*, *<y>* and *<z>* indicates the coordinate from the center of the light. By specifying the values *<ry>* and *<rz>*, the rotation of the light around the y- and z-axis is defined. The values *<w>* and *<h>* indicates its width and height.

Please note, that every light except for the brake and reversing light must be triggered by the DVA-mechanism, using the quantities *DM.Lights.\**. For further information regarding this topic, please have a look at [section 6.3 'DVA: Online Manipulation of the Simulation' on page 357](#).

If you would like to round the light, this can be realized with the rounding coefficient. A value of 0 indicates a squared light, while a value of 1 supplies a rounded one.



A rounding coefficient of 1 does not lead to a circle light automatically. To realize this, the values *<w>* and *<h>* must be equal.

#### Example

```
### BEGIN IPG-MOVIE-INFO
# Light Brake 0.265 0.595 0.735 44 -15 0.22 0.23 .5
# Light Brake 0.265 -0.595 0.745 44 15 0.22 0.23 .5
# Light Brake 0.424 0.000 0.975 50 0 0.31 0.05 0
### END IPG-MOVIE-INFO
```



As an example take the TestRun Examples > VehicleDynamics > Braking.



Figure 7.48: Flashing brake lights

## 7.4.2 Integrating mtex-Files

Instead of inserting a simple \*.jpg or \*.png file into your simulation, you can select a \*.mtex file which describes the respective surface properties. The \*.mtex-file is in the usual Car-Maker infofile format.

Within this \*.mtex file you can define a so-called ReflectionMap, which is used to control the light influences on a reflective surface. The \*.mtex-file can contain different values with different optical effects. If no \*.mtex-file is used to display a texture on the road, there is no light influence on it.

As an example, you can insert puddles, which have a very reflective surface - check the file *Movie/Textures/Puddles/puddles03.mtex* to get an impression of this.

Other usage areas for mtex-files are, for example, road paintings. You can control the amount of horizontal and vertical repetition of the texture to get a realistic look. As an example for this, check the file *Movie/Textures/painting.mtex*.

**Example File** This is the content of an \*.mtex-file:

```
#INFOFILE1.1 - Do not remove this line!
FileIdent = IPGMovie-TextureInfo
FileType = MovieTextureInfo
Texture.DiffuseMap = Textures/LaneArrow_StraightRight.png
Texture.Repeat.Horizontal = 2
Texture.Repeat.Vertical = 3
```

Within the \*.mtex-file, you have the following possibilities to define the surface properties:

*Texture.ReflectionMap* = Enables the reflection on the texture

*Texture.DiffuseMap* = With this option, the texture is displayed as usual

*Texture.Repeat.Horizontal* = X - With this option, the texture is repeated horizontally. X stands for the number of repetitions.

*Texture.Repeat.Vertical* = X - With this option, the texture is repeated vertically. X stands for the number of repetitions.

*Specularity*= Value1 (red, range from 0-1) Value2 (green, range from 0-1) Value3 (blue, range from 0-1) Value4 (Retroreflection, range from 1=no reflection - 0= full reflection), Value5 (gloss level, range from 1 (selective) - 128 (extensive))

## 7.5 Active Lights

### 7.5.1 Introduction

The goal of this chapter is to explain the steps required to setup the active lights in the TestRun mentioned below, and in this process, to help to understand the underlying mechanisms.

- TestRun: *Examples/CarMakerFunctions/IPGMovie/ActiveLights*

Please note, that the *Active Lights* functionality of IPGMovie makes high demands on the graphical system. We recommend to use NVidia/AMD graphic cards which are not older than two years.

### 7.5.2 General information

#### What is an Active Light?

In IPGMovie, there is already a menu item *View > Show > Lights*. This option displays elements that *look* like functional lights: they are very efficient performance-wise, but do not actually cast a light. In order to actually cast a light, we need a different sort of lights: active lights.

Both types of lights work together very well: the inactive light displays a change in the light state, and the active light actually lights the scene.

IPGMovie uses a lighting technique to display the active lights that is called *per-pixel lighting*, which is achieved through an OpenGL technology called *shaders*.

#### Available Types of Light

- Global light
  - This is the global light for the scene. It can be changed with the environment settings *Scene > Background*. Consider this as the sun light (or moon light). In the OpenGL terminology, this is a *directional* light.
- Spotlight
  - This is an ideal spotlight. It's a cone of light, with a position, a direction, the angle defining the opening of the cone, whose intensity declines with the distance.
- Positional

- This is an ideal light bulb. It's a single source of light, which intensity declines with the distance.
- Also known as "Point light".
- Map
  - This is a more realistic light. It is based on a spotlight, with the intensity at a given distance being additionally affected by a so-called *Iso Candela Maps* (from now on abbreviated as *ICM/ILM*). This diagram must first be converted into a greyscale PNG / JPG image, that is used as a lookup texture. IPGMovie provides a converter, launched in the menu *File*.
- MaskMap
  - This is an extension of the *Map* light, where a mask (greyscale PNG/JPG as well) can be given additionally to the Iso Lux/Iso Candela Maps.

Not all types of lights are available for all objects in the scene:

Table 7.5: Available types of light over objects in scene

Object	Spotlight	Positional	Map	MaskMap
MovieGeo marker on the road	X			
Traffic objects			X	X
Test vehicle		X	X	X
Trailers		X		

## Impact on Performances

The display of the realistic lighting will usually be heavier on the graphic hardware, possibly resulting in slower performances. You will find more detailed information regarding the simulation performance within the following chapter.

### Amount and type of lights



The more active lights you have in your scene, the slower the display will be, since the effect of each light needs to be computed for almost each point in the scene.

Besides the fact that there can be only one global light, each type of light can have at most 8 occurrences, with the additional limit that the grand total of lights cannot be higher than 20.

Another limit is that only 3 traffic objects may have lights.

For instance:

- you cannot have more than 8 Map lights
- if you already have 4 Positional lights, 4 Map lights and 8 Spotlights defined, all over your scene, you have only 4 active lights left

### Lights authoritatively turned off

In order to save processing power, IPGMovie will authoritatively turn some lights off, based on the following criteria:

- Road lights
  - If the light, within a radius of 50m, is not visible on screen or the distance between the light and the camera is greater than 1250m, it will be turned off.
- Traffic objects with lights
  - If the light, within a radius of 220m, is not visible on screen or the distance between the light and the camera is greater than 1500m, it will be turned off.

- If the distance between the traffic object and the test vehicle is greater than 1000m, its lights are turned off

## Special View Modes of IPGMovie

Active lights will also be active in the VDS stream/preview, as well as in the fisheye lens modes.

### 7.5.3 Active Lights on the Road

The road can currently only have spotlights.

#### Add Street Lights

After defining the road segments, let us add a few street lights, which will use spotlights. To positionate the lights, we will use MovieGeo markers.

#### Light Definition file

Lights are defined with the help of text files (Infofiles), which follow a simple format:

```
Key = Value
MultilineKey:
Value1
Value2
```

Lines starting with # are commented out.

Let us have a look at:

```
<Installation-directory>/Movie/Misc/Streetlight.activelights
```

```
Lights.Amount = 1
Light0.Type = Spotlight
Light0.Position = .21 2.1 4.75
Light0.Color = 1.0 1.0 1.0
Light0.Dir = 0 0 -1.1
Light0.Cutoff = 30
Light0.Exponent = 15
Light0.Attenuation = .1 .2 .3
Light0.Condition = 1
```

The file declares 1 light of the type `Spotlight`. Inside the definition file this light will be referred to as `Light0`.



`Lights.Amount` declares, how many lights are defined in the file, not globally. All light definitions start with the definition of `Light0`. Indeed, the index `N` for `Light<N>` starts at 0 and 2) is reinitialized for each light definition file. In other words: in each light definition file, the first light defined will be called `Light0`.

The `Position` of the light is not absolute to the road, but relative to the position of the MovieGeo marker to which it is attached (see [Figure 7.49](#)).

The `Color` component is given with the usual R G B values.

`Dir` is the direction of the light, in this case we point vertically to the ground.

*Cutoff* is the angle defining the width of the light cone. It cannot be bigger than 90° degrees.

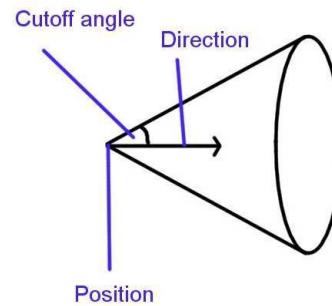


Figure 7.49: Position and Orientation of an active light

*Exponent* expresses how focused around the direction the light beam is.

*Attenuation* is the list of the constant, the linear and the quadratic attenuation factors respectively. Those factors influence the intensity of the light, as a function of the distance between the light source and the point in 3D space for which we are computing the illumination.

$$f(\text{dist}) = \min\left(\frac{1}{(\text{const} + \text{dist} \times \text{linear} + \text{dist}^2 \times \text{quadratic})}, 1.0\right) \quad (\text{EQ 4})$$

*Condition* is a boolean test (i.e a check that should amount to TRUE or FALSE) that controls whether the light is turned *on* or *off*. In this case the light is always on.

### OBJ File

We will use a simple OBJ file to refer to both the 3D model of the street lamp and the light definition file.

Let us have a look at *Movie/Misc/Streetlight\_Light.obj*.

```
### BEGIN IPG-MOVIE-INFO
# Include StreetLight_Light.mobj
# ActiveLights Streetlight.activelights
### END IPG-MOVIE-INFO
```

It is very simple: we just need to reference

- the 3D model (# *Include StreetLight\_Light.mobj*)
- the light definition file (# *ActiveLights Streetlight.activelights*)

The file *Streetlight\_Light.mobj* is also available in *Movie/Misc/*, but it could be stored somewhere else, as long as it can be found by IPGMovie.

The only thing to do now is to set the geometry file of the MovieGeo road marker to *Misc/Streetlight\_Light.obj*.

To add turned off street lamps, just use a 3D model without an active light as a MovieGeo marker.

## 7.5.4 Active Lights on the Test Vehicle

### Add lights

#### Light Definition File

Let us have a look at:

## OBJ file

Please refer to file:

*Movie/3DObjects/Vehicles/ActiveLights/VW\_Beetle\_2012\_Lights.obj*

Its structure is simple. It references the main 3D model, the inactive lights to display and the active light definition file.

The interesting part is the position of `# Subscribe`. This is how we will get the necessary quantities for the control of the lights: to control the mode, we will use the quantities `VC.Lights.*`. The Quantity `Lights.Mask` is a custom, user-created quantity that will be used to switch on using a light mask. Refer to [section 7.5.5 'Control the Active Lights' on page 441](#) for more details.

The quantities `Beam.*`, to control the orientation are automatically subscribed. Indeed, the prefix defined in `LightX.Control.Quantity` will be used to automatically generate and subscribe the necessary Quantities.

```
### BEGIN IPG-MOVIE-INFO
# Include VW_Beetle_2012_Lights.obj
# Light Brake    0.270  0.595  0.740  44   -15   .22  .23   1
# Light Brake    0.270  -0.595  0.740  44    15   .22  .23   1
# Light Brake    0.424  0.000  0.975  50     0   .31  .05   1
# Light IndicatorL 0.31   0.605  0.785  44   -15   .17  .09   1
# Light IndicatorL 3.90   0.705  0.450   0  -135   .22  .06   1
# Light IndicatorR 0.31  -0.605  0.785  44    15   .17  .09   1
# Light IndicatorR 3.90  -0.705  0.450   0  135   .22  .06   1
# ActiveLights VW_Beetle_2012.activelights
# Subscribe VC.Lights.Brake VC.Lights.IndL VC.Lights.IndR VC.Lights.LowBeam
VC.Lights.HighBeam
# Subscribe Lights.Mask
### END IPG-MOVIE-INFO
```

The advantages of using a scheme like this to define lights:

- 1) while you are in ABRAXAS mode, you still get the active lights.
- 2) you can modify the light definition file (e.g. map textures or color of lights) and simply reload the geometry (right mouse click > *Reload Geometry*) to apply the changes, you do not need to restart the simulation.

## Vehicle Dataset

What we did here is also very simple:

- Load the vehicle data set DemoCar
- Change the Movie geometry (tab *Misc*) to *VW\_Beetle\_2012\_Lights.obj*
- Save the modified data set under the name *VW\_Beetle\_2012.activelights*

## Trailer

For a trailer, the principle is exactly the same:

- Create the light definition file. Remember that only positional lights are allowed for trailers.
- Create an OBJ file that will reference both the 3D model and the light definition file.
- Use that OBJ file as the Movie Geometry for the trailer data set.
- Save the modified dataset under a new name, and use it.

- The Position is absolute neither to the road nor to the vehicle but relative to the body of the vehicle to which the light is attached.
- Control is a boolean (0 mean False and 1 means True) that indicates whether the lights can be controlled by using the quantity mentioned in Control.Quantity.
- Control.Quantity is the root name (prefix) of a family of quantities that can be used to control the light. Those Quantities are automatically subscribed to when starting the simulation. In this case, the value BeamL means that...
  - ... the quantities BeamL.rx BeamL.ry BeamL.rz will be used to rotate the light around the X, Y and Z axis. Those quantities express angles in radian.
  - ... the quantity BeamL.mult will be used to control the intensity of the light.



You can choose which prefix you want, but the resulting quantities must be subscribed to in the OBJ file and must be initialized in the first maneuver (see [section 7.5.5 'Control the Active Lights' on page 441](#)).

- Color is as described in [section 'Light Definition File' on page 435](#).
- Area is the color for the area. See [section 7.5.7 'Using Active Lights' on page 444](#).
- Dir is a vector giving the direction of the light source from its position and Up is a vector giving the upward direction. Typically Up and Dir will be perpendicular in the plane XZ. Those two vectors are needed to project the ICM/ILM texture onto the polygon being rendered.
- the attenuation is not controllable, it is computed as follows (where dist is the distance between the light source and the point in 3D space for which we are computing the illumination).

$$\text{attenuation} = \min\left(\frac{1}{dist^2}, 1.0\right) \quad (\text{EQ } 5)$$

- Functions is a multiline key that describes the different functions or modes of the light. This is used for instance to describe the low beam, the high beam, etc. For a detailed description please read [section 'Functions for MaskMap and Map lights' on page 437](#).

Let us have a closer look at Light3:

- the Position is absolute neither to the road nor to the vehicle but relative to the body of the vehicle to which the light is attached.
- Color and attenuation: as described in [section 'Light Definition file' on page 434](#).
- the Condition simply tests if the brake or the turn indicator are active. The used quantities are subscribed to by Movie because of the line # Subscribe in the OBJ file. Positional light and spotlights cannot have several modes, they can only be off or on.

### Functions for MaskMap and Map lights

You can have as many functions as you want, but out of performance concerns you should avoid having more than 10.

Most of the time, one function will be activated. This means that either the light is off, or it is on using one of the modes described.

The functions are defined as follows:

- Map light  
`<Condition> <Coverage> <Baseintensity> <Map texture>`
- MaskMap light  
`<Condition> <Coverage> <Baseintensity> <Map texture> <Mask texture>`

**Condition** As already explained, this is a boolean test that will determine when the function is used. Consider the condition to be the body of an `if` statement.

To determine which function should be used, all listed conditions are evaluated, until one of them is evaluated to be true. This condition will be used. If no conditions are evaluated to be true, the light is off. This means, that the order in which the functions are listed is VERY important.

Have a look at [section 7.5.5 'Control the Active Lights' on page 441](#) to find out how to switch from one function to another.

The syntax for a condition is very simple and based on the syntax of *TCL* and its supported mathematical functions, but it will also be familiar to users of any C-based programming language.

- the condition must not contain any spaces
- to indicate a quantity, enclose it in `$Qu()`, for instance `$Qu(VC.Lights.HighBeam)`
- all usual arithmetical operations: `+`, `-`, `*`, `/`, `%` (modulo), `**` (power)
- all usual comparison operations: `<`, `>`, `<=`, `>=`, `==` (equality), `!=` (difference)
- all usual logic operations: `&&` (logical AND), `||` (logical OR), `!` (logical NOT)
- the usual two way conditional switch: `x?y:z` (if `x` is evaluated to be true, the `y` else `z`)
- all usual bitwise operations: `~, &, |, ^, <<, >>`
- the following mathematical functions: `abs`, `acos`, `asin`, `atan`, `atan2`, `bool`, `ceil`, `cos`, `cosh`, `double`, `entier`, `exp`, `floor`, `fmod`, `hypot`, `int`, `isqrt`, `log`, `log10`, `max`, `min`, `pow`, `rand`, `round`, `sin`, `sinh`, `sqrt`, `rand`, `tan`, `tanh` `arg`, `wide`

If you make a mistake in the syntax of the condition, you will receive an error message.

**Coverage** This is a set of 4 values, usually given when converting data from an ICM/ILM to a PNG/JPG texture.

- maximum horizontal angle (left)
- minimum horizontal angle (right)
- maximum vertical angle (top)
- minimum vertical angle (bottom)

Those parameters must follow the rules:

- *left* and *right* angle must be smaller than or equal to 90 degrees
- *left* must be equal to *right*
- *top* and *bottom* must be smaller than or equal to *left*

For more information have a look at [section 'Iso Candela Maps' on page 444](#).

**Base intensity** This is a base multiplier for the intensity defined in the map texture.

**Map texture** This is the PNG/JPG file name of the picture made from the ICM data. We recommend using the PNG format to avoid compression artefacts. The files are found by IPGMovie like any other texture file. We recommend to store them in the folder *Movie* of your project directory. If the file is stored in a subfolder of the folder *Movie*, but not in the same folder as the light definition file, you will need to include the relative path name before the file name.

Examples (let us consider that the light definition file is stored in `<Project directory>/Movie/Lights`)

Table 7.6: Example of relative map texture directory

Map texture file	Valid light function definition
<code>&lt;Project folder&gt;/Movie/Map.png</code>	<code>1 45 -45 15 -15 100 Map.png</code>
<code>&lt;Project folder&gt;/Movie/Lights/Map.png</code>	<code>1 45 -45 15 -15 100 Map.png</code>
<code>&lt;Project folder&gt;/Movie/MyTextures/Map.png</code>	<code>1 45 -45 15 -15 100 MyTextures/Map.png</code>

IPGMovie has a few ready-made generic textures available:

Table 7.7: Generic map textures

File name	Function	Vehicle type
<code>IPG_ICM_LB_default.png</code>	Low beam	Car & Truck
<code>IPG_ICM_HB_default.png</code>	High Beam	Car & Truck
<code>IPG_ICM_LB+HB_default.png</code>	Low and High beams	Car & Truck

There are no ready-made map textures for Motorcycles yet.

Currently only greyscale textures are supported.

- Mask texture** A \*.png/\*jpeg file that defines a mask to additionally apply to the map. This must be a greyscale texture. Again, we recommend to use the \*.png-format.
- Basically both map and mask textures will be multiplied together. That means that all black areas (RGB = 0,0,0) will result in hiding those areas in the map, while all white areas (RGB = 1,1,1) will keep the map as it is. Nuances of grey will darken the map accordingly.
- Have a look at the mask texture `Movie/3DObjects/Vehicles/ActiveLights/` `Mask.jpg`. When you run the TestRun `CarMakerFunctions/IPGMovie/ActiveLights`, a mask will be in effect at 125.0s.



Figure 7.50:  
Left picture: Maneuver #10 - Function#3 on: LowBeam on, no mask.  
Middle picture: the mask to be applied  
Right picture: Maneuver #11 - Function#1 on: LowBeam on, mask active

A mask texture file name is optional. If it is missing, the light will behave like a Map light. You can use this for your advantage.

- If you need more than 8 map lights: simply use MaskMap lights without giving a Map texture

- To turn the light on for a light function where no mask is needed. Example:

```
Light0.Functions:
    {int($Qu(LightL.Mask))==1} 35 -35 3 -12 150 {IPG_ICM_LB_default.png}
    {int($Qu(LightL.Mask))==2} 35 -35 3 -12 150 {IPG_ICM_LB_default.png} ActiveLights/
Mask1.jpg
    {int($Qu(LightL.Mask))==3} 35 -35 3 -12 150 {IPG_ICM_LB_default.png} ActiveLights/
Mask2.jpg
    {int($Qu(LightL.Mask))==4} 35 -35 3 -12 150 {IPG_ICM_LB_default.png} ActiveLights/
Mask4.jpg
```

### OBJ file

Please refer to file:

*Movie/3DObjects/Vehicles/ActiveLights/VW\_Beetle\_2012\_Lights.obj*

It references the main 3D model, the inactive lights to display and the active light definition file.

The interesting part is the position of # Subscribe. This is how we will get the necessary quantities for the control of the lights: to control the mode, we will use the quantities *VC.Lights.\**. The Quantity *Lights.Mask* is a custom, user-created quantity that will be used to switch on using a light mask. Refer to [section 7.5.5 'Control the Active Lights'](#) for more details.

The quantities *Beam.\**, to control the orientation are automatically subscribed. Indeed, the prefix defined in *LightX.Control.Quantity* will be used to automatically generate and subscribe the necessary Quantities.

```
### BEGIN IPG-MOVIE-INFO
# Include VW_Beetle_2012_Lights.obj
# Light Brake      0.270  0.595  0.740  44   -15   .22  .23   1
# Light Brake      0.270 -0.595  0.740  44    15   .22  .23   1
# Light Brake      0.424  0.000  0.975  50     0   .31  .05   1
# Light IndicatorL 0.31   0.605  0.785  44   -15   .17  .09   1
# Light IndicatorL 3.90   0.705  0.450  0   -135   .22  .06   1
# Light IndicatorR 0.31  -0.605  0.785  44    15   .17  .09   1
# Light IndicatorR 3.90  -0.705  0.450  0    135   .22  .06   1
# ActiveLights VW_Beetle_2012.activelights
# Subscribe VC.Lights.Brake VC.Lights.IndL VC.Lights.IndR VC.Lights.LowBeam
VC.Lights.HighBeam
# Subscribe Lights.Mask
### END IPG-MOVIE-INFO
```

The advantages of using a scheme like this to define lights:

- 1) while you are in ABRAXAS mode, you still get the active lights.
- 2) you can modify the light definition file (e.g. map textures or color of lights) and simply reload the geometry (right mouse click > *Reload Geometry*) to apply the changes, you do not need to restart the simulation.

### Vehicle Dataset

What we did here is also very simple:

- Load the vehicle data set DemoCar
- Change the Movie geometry (tab *Misc*) to *VW\_Beetle\_2012\_Lights.obj*
- Save the modified data set under the name *VW\_Beetle\_2012.activelights*

### Trailer

For a trailer, the principle is exactly the same:

- Create the light definition file. Remember that only positional lights are allowed for trailers.
- Create an OBJ file that will reference both the 3D model and the light definition file.
- Use that OBJ file as the Movie Geometry for the trailer data set.
- Save the modified dataset under a new name, and use it.

## 7.5.5 Control the Active Lights

### Turn the Lights on/off - Switch Light Modes

Let us go back to the file:

*Movie/3DObjects/Vehicles/ActiveLights/VW\_Beetle\_2012.activelights*

The functions defined for Light0 and Light1 are:

```
{$Qu(VC.Lights.LowBeam)>0&&$Qu(Lights.Mask)>0} 43 -43 3 -12 150 IPG_ICM_LB_default.png
ActiveLights/Mask.jpg
{$Qu(VC.Lights.LowBeam)&&$Qu(VC.Lights.HighBeam)} 43 -43 12 -12 275 IPG_ICM_LB+HB_default.png
{$Qu(VC.Lights.LowBeam)}
43 -43 3 -12 150 IPG_ICM_LB_default.png
{$Qu(VC.Lights.HighBeam)}
43 -43 9 -4 275 IPG_ICM_HB_default.png
```

Reminder:

- The simulation program automatically sets the quantities *VC.Lights.\** according to the quantities *DM.Lights.\**.
- All quantities are subscribed to by IPGMovie thanks to the lines # Subscribe in the OBJ file.

Therefore, to change between modes/functions, you simply need to modify the quantities either with the minimaneuver language or with Realtime Expressions.

Let us check some maneuvers out of the TestRun *CarMakerFunctions/IPGMovie/Activelights*:

Table 7.8: Example of minimaneuver commands controlling lights

Maneuver	Minimaneuver Commands	Result
0,13	DVAwr DM.Lights.MainLight Abs -1 0 DVAwr DM.Lights.HighBeam Abs -1 0 DVAwr DM.Lights.Indicator Abs -1 0 DVAwr DM.Lights.Hazard Abs -1 0 Eval Qu::Lights.Mask = 0	Initialization, all lights off, no mask
1	DVAwr DM.Lights.MainLight Abs -1 2 DVAwr DM.Lights.HighBeam Abs -1 1	Low beam on, high beam on => Function #2 is on
2	DVAwr DM.Lights.MainLight Abs -1 0 DVAwr DM.Lights.HighBeam Abs -1 1	Low beam off, high beam on => Function #4 is on
3	DVAwr DM.Lights.MainLight Abs -1 2 DVAwr DM.Lights.HighBeam Abs -1 0	Low beam on, high beam off => Function #3 is on
7	DVAwr DM.Lights.MainLight Abs -1 2 DVAwr DM.Lights.HighBeam Abs -1 1	Low beam on, high beam on => Function #2 is on
10	DVAwr DM.Lights.HighBeam Abs -1 0	Low beam still on, high beam off => Function #3 is on
11	Eval Qu::Lights.Mask = 1	Low beam still on, Lights.Mask > 0 => Function #1 is on
12	Eval Qu::Lights.Mask = 0	Low beam still on, no mask => Function #3 is on



The initialization `Eval Qu::Lights.Mask = 0` is absolutely necessary, otherwise you will receive an error message from IPGMovie: “*Missing Quantities! Needed for animation, but not provided.*”.

DVAwr is a minimaneuver command and Eval marks a Realtime Expression, both are described in the User’s Guide.

### Rotate the Lights Automatically - Simple Adaptive Lights

The principle is exactly the same, but this time we will use the quantities BeamR.\* BeamL.\* Let us check some maneuvers out of the TestRun *CarMakerFunctions/IPGMovie/ActiveLights*:

Table 7.9: Example of simple adaptive lights with minimaneuver commands

Maneuver	Minimaneuver Commands	Result
0	<code>Eval Qu::BeamL.rz = 0</code> <code>Eval Qu::BeamL.ry = 0.025</code> <code>Eval Qu::BeamL.rx = 0</code> <code>Eval Qu::BeamL.mult = 1.0</code> <code>Eval Qu::BeamR.rz = 0</code> <code>Eval Qu::BeamR.ry = 0.025</code> <code>Eval Qu::BeamR.rx = 0</code> <code>Eval Qu::BeamR.mult = 1.0</code>	Initialization, lights slightly (0.025 rad = 1.5°) pitched toward the ground
8	<code>Eval Qu::BeamL.rz = Steer.WhlAng*0.334</code> <code>Eval Qu::BeamR.rz = Steer.WhlAng*0.334</code>	Very primitive adaptive lights: we use the steering wheel angle to determine the yaw of the lights
9	<code>Eval Qu::BeamL.rz = 0</code> <code>Eval Qu::BeamR.rz = 0</code>	Turn off the adaptive lights.

#### The initialization

```
Eval Qu::BeamL.rz = 0
Eval Qu::BeamL.ry = 0.025
Eval Qu::BeamL.rx = 0
Eval Qu::BeamL.mult = 1.0
Eval Qu::BeamR.rz = 0
Eval Qu::BeamR.ry = 0.025
Eval Qu::BeamR.rx = 0
Eval Qu::BeamR.mult = 1.0
```

is absolutely necessary (no matter which values are given), otherwise you will receive an error message: “*Missing Quantities! Needed for animation, but not provided.*”.

## 7.5.6 Active Lights on Traffic Objects

### Simple Lights

#### Light Definition File

Let us have a look at *Movie/Examples/ActiveLights/Audi\_TT\_1998.activelights*.

```
Lights.Amount = 2
#-----#
Light0.Type = MaskMap
Light0.Position = 3.7 0.65 .725
Light0.Control = 0
Light0.Control.Quantity =
# {1 0 0} with ry = 2
Light0.Dir = 0.99939 0.0 -0.03490
# {0 0 1} with ry = 2
```

```

Light0.Up =          0.03490 0.0 0.99939
Light0.Color = 1 1 1
Light0.Color.Area = 1 0 0
Light0.Functions:
  1 43 -43 3 -12 150 IPG_ICM_LB_default.png
#-----#
Light1.Type = MaskMap
Light1.Position = 3.7 -0.65 .725
Light1.Control = 0
Light1.Control.Quantity =
Light1.Dir = 0.99939 0.0 -0.03490
Light1.Up = 0.03490 0.0 0.99939
Light1.Color = 1 1 1
Light1.Color.Area = 0 0 1
Light1.Functions:
  Light0

```

The content of this file is very similar to the light definition for the test vehicle, as presented in [section 'Light Definition file' on page 434](#).

- `Light0.Control = 0` means that we do not wish to control the lights with quantities. It is possible though, and works as described in [section 7.5.4 'Active Lights on the Test Vehicle' on page 435](#).
- The *Dir* and *Up* vectors are already rotated around the Y axis by about 2 degrees, toward the ground.
- A single *function* is defined: the low beam is always on (unless the criteria described in apply) since the *Condition* given (1) will always evaluate to TRUE.

## OBJ File

Let us have a look at:

`Movie/3DObjects/Vehicles/ActiveLights/Audi_TT_1998_Lights.obj`

```

### BEGIN IPG-MOVIE-INFO
# Include Audi_TT_1998.mobj
# Wheel.fl 3.15 0.72 0.30 0.31 0.18
# Wheel.fr 3.15 -.72 0.30 0.31 0.18
# Wheel.rl 0.72 0.72 0.30 0.31 0.18
# Wheel.rr 0.72 -.72 0.30 0.31 0.18
# ActiveLights Audi_TT_1998.activelights
### END IPG-MOVIE-INFO

```

We reference the 3D model to use (`# Include Audi_TT_1998.mobj`), make sure it has wheels when used as a traffic object and reference the light definition file (`# ActiveLights Audi_TT_1998.activelights`).

The only thing left to do is to use this geometry file as *Movie Geometry* for the appropriate traffic object(s) in the traffic editor.

Since we are dealing with a traffic object, all we need are those two files (OBJ and light definition file), a new vehicle dataset is not required.

## Automatic Mode Change

Basically, everything that can be done with light functions for the test vehicle can also be done for the lights of traffic objects:

- Controlling the lights with quantities
  - Light definition file: use `Light<N>.Control` and `Light<N>.Control.Quantity`
  - OBJ file: subscribe to the appropriate quantities
  - Maneuvers: use Realtime Expressions to set the values of the quantities

- Changing light modes by declaring several functions

### Light definition file

Let us have the traffic object switch from high+low beam to low beam when it is close enough to the test vehicle.

This is achieved with the help of two simple functions:

```
Light0.Functions:  
    (abs($Qu(Traffic.Lights_A.sRoad)-$Qu(Vhcl.sRoad))<100)    43    -43     3    -12    150  
IPG_ICM_LB_default.png  
    1  
        43    -43     12    -12    275  
IPG_ICM_LB+HB_default.png
```

and

```
Light1.Functions:  
    Light0
```

What we describe is:

- either the traffic object Audi\_HLA is close enough (100 m road distance) to the test vehicle, then we use the low beam function (see [Table 7.7: Generic map textures](#))
- or we use the high+low beam function (see [Table 7.7: Generic map textures](#))

The rest of the file Movie/Examples/ActiveLights/Audi\_TT\_1998\_Active.activelights is the same as the one described in [section 'Light Definition file' on page 434](#).

Of course, if the criteria described in [section 'Lights authoritatively turned off'](#) apply, the lights will be turned off.

### OBJ file

The content of the file Movie/3DObjects/Vehicles/ActiveLights/Audi\_TT\_1998\_Active\_Lights.obj is very similar to the one described in , with the only difference being that another light definition file is used (Movie/3DObjects/Vehicles/ActiveLights/Audi\_TT\_1998\_Active.activelights).

Again, this file should be used as the *Movie Geometry* in for the appropriate traffic object in the traffic editor.

## 7.5.7 Using Active Lights

Simply start IPGMovie and the simulation.

The active lights will be automatically enabled if light definition files are provided in the OBJ files, as explained above.

If active lights are enabled, IPGMovie alters the material properties of traffic signs, guide posts, number plates and pylons in a way that they show retroreflective behavior.

For \*.obj files we provide a custom property Rr for corresponding material definitions. If the Rr value is set to 1, the alpha value given for the specularity term defines the amount of retroreflectivity. The actual amount of specularity is then defined as 1 - alpha. As usual, the given rgb-values define the reflection coefficients for the different color channels.

### Iso Candela Maps

Most of the 2D mapping data for Iso Candela Maps is available in text form. IPGMovie currently requires this data as a 2D texture. The conversion will give you essential meta-data for the light definition files.

Let us check again the file VW\_Beetle\_2012.activelights:

### Light0.Functions:

```
( $Qu(VC.Lights.LowBeam)>0 && $Qu(Lights.Mask)>0)      43 -43 3 -12 85  IPG_ICM_LB_default.png
Examples/ActiveLights/Mask.jpg
( $Qu(VC.Lights.LowBeam) && $Qu(VC.Lights.HighBeam) )   43 -43 12 -12 185
IPG_ICM_LB+HB_default.png
$Qu(VC.Lights.LowBeam)                                     43 -43 3 -12 85  IPG_ICM_LB_default.png
$Qu(VC.Lights.HighBeam)                                    43 -43 9 -4 185  IPG_ICM_HB_default.png
```

The values "43 -43 3 -12" describe the angular coverage of the mapping:

- vertically: from 43 degrees (left) to -43 degrees (right)
- horizontally: from 3 degrees (top) to -12 degrees (bottom)

Those values will be displayed on the screen and saved in an accompanying text file by the conversion tool.

### Converting the data

- 1) Open the conversion tool via *File > Iso-Candela Converter...*

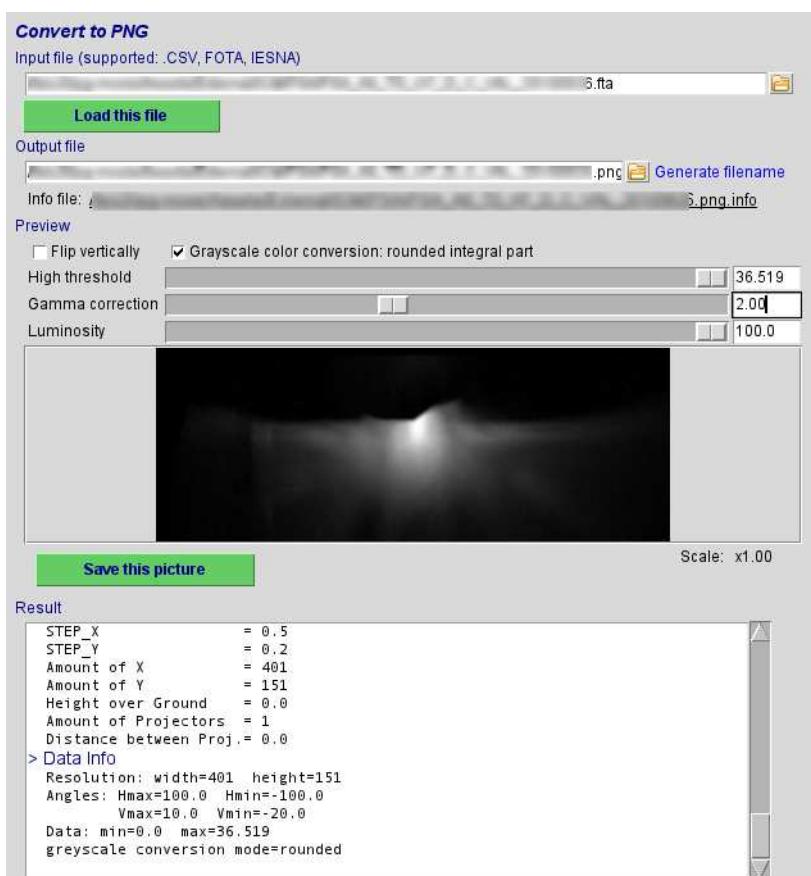


Figure 7.51: Convert to PNG - GUI

- 2) Select the input file by clicking on *Load this file...*

Supported formats are *FOTA* (file extension: FTA), *IESNA* (file extension: IES) and *CSV*. Using the CSV format, you will need to provide column and row where the data starts: simply locate the row and the column where the angles are given. As far as the rounding option is concerned, you can leave it as is.

The output file name can be generated automatically if you click on *Generate filename*. The name of the info file is derived from the output file name (see below for more information about the output).

The result of the conversion is shown in the textfield on the bottom.

### 3) Modify the picture.

Use the sliders to get an appropriate result in IPGMovie. You will usually only need to modify the *gamma* and the *luminosity* settings.

The preview will be updated accordingly.

### 4) Save

Click on *Save this picture* to create the PNG file.

### Output files

There are two outputs:

On the screen, a text field on the bottom will provide information about the file to convert, about the conversion and about the file converted. Very important is the angular coverage (Hmin, Hmax, VMin and Vmax). You will need this for the light definition files, as explained before in [section 7.5.4 'Active Lights on the Test Vehicle' on page 435](#).

A companion text file is written alongside the PNG file, which contains the conversion information shown last in the text field. If the PNG file is named ABC.png, then the companion text file will be named ABC.png.txt.

## 7.6 Pedestrians

### 7.6.1 Basic technical information about animated characters in general

An animated character is generally a combination of three components.

- a 3D-Model, with materials (usually one or more textures)
- a skeleton (a set of weighted joints, linked together)
- predefined animation(s).  
Most animations are based on motion captured from real persons.

IPGMovie adds a fourth: the data set, and 3D-models, materials and skeleton are all packed in our *Source Asset*. Other than that, the same principles apply everywhere.

A simplified description of what happens when a pedestrian moves is given here:

- The animation tracks (from the animation files) define a time-based specific position for each joint of the skeleton.
- The skeleton is then repositioned according to the animation track(s) and the current time.
- Finally, the geometry of the 3D-model is modified in real-time according to the modified skeleton.
- The model is rendered on the screen, using the modified geometry and the materials (base colors, usually given through texture files).

## 7.6.2 Available characters

There are currently 4 available models, some of them offering several variations.

In total, we provide 9 predefined pedestrians.

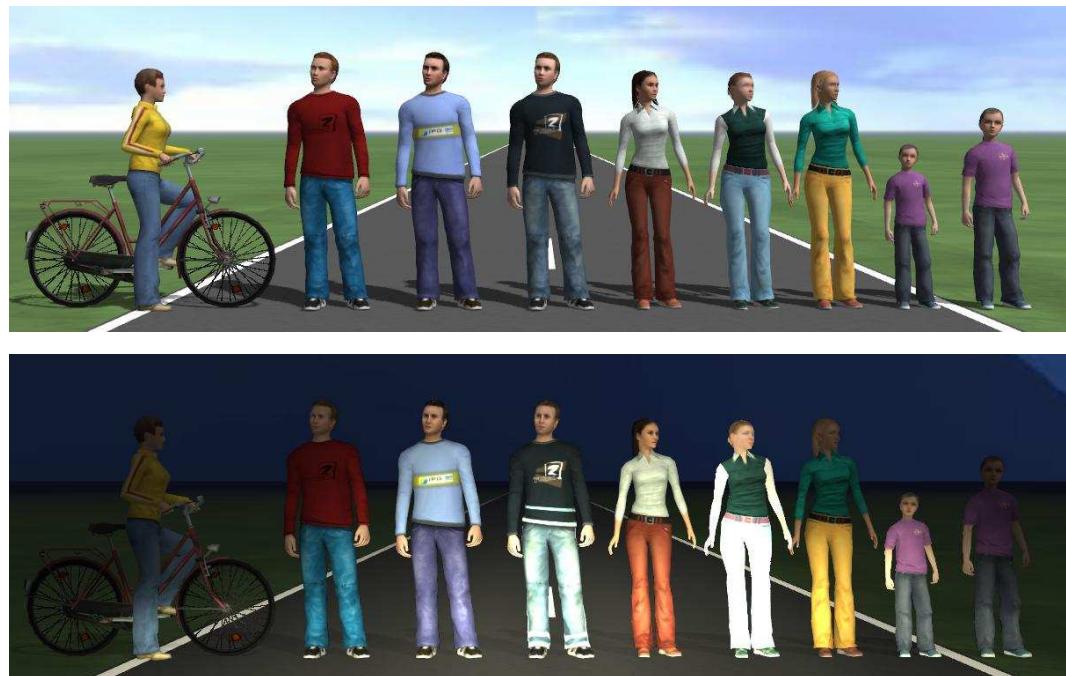


Figure 7.52: Available pedestrian models

- 1 Male
  - Clothing variation #1
  - Clothing variation #1, with retroreflexive stripes
  - Clothing variation #2
- 1 Female
  - Clothing variation #1
  - Clothing variation #2
  - Clothing variation #3
- 1 Female Biker
- 1 Child
  - Height: 142cm
  - Height: 115cm

Characters with retroreflexive clothing work only when used in a testrun with vehicles supporting active lights. Please refer to the testrun *Examples/CarMakerFunctions/IPGMovie/Pedestrian* for an example.

## 7.6.3 Usage as traffic object

A pedestrian is to be used as a normal traffic object.

- Choose an appropriate model for the field Movie Geometry
  - All predefined pedestrians are so far only available in *3DObjects/Characters*

- Define a set of appropriate maneuvers.

Please load the test run *Examples/CarMakerFunctions/IPGMovie/Pedestrian* to see an example.

Which animation (stand, walk or run) is used to give the pedestrian a natural motion is based on the speed of the traffic object. Therefore to get best results, you should choose realistic speeds when defining the maneuvers.

#### 7.6.4 File types and folder structure

A pedestrian is organized in a way similar to the ego vehicle:

- a dataset, referencing a geometry file and a few specificities
- a source asset, as a set of files
- additional files needed to run the simulation - in our case: the animation files

Comparison with the vehicle data sets: for instance, you can have several separate data sets for a vehicle, all using the same *Movie Geometry*, but defining different engine configurations. The *Movie Geometry* is then also a set of files: OBJ and MTL files for the carosserie, 3d-models of the wheels, light configuration file, etc..

The concept is similar for the pedestrians.

For instance, the two children use the same source asset, but each dataset defines a different scale.

##### Data set

This is your entry point when you choose a pedestrian. This is where and how a specific pedestrian is set up, typically based on

- a mandatory source asset (the geometry files)
- which version to use, among possibly several available in the asset
- scale

If no version is defined, the first one in the source asset will be used.

If the scale is not given, or if its value is 0.0, then the default scale defined in the source asset will be used.

These are the usual settings, but an extended configuration (like which texture to use for the retroreflexive stripes on the clothing) is also available.

A data set can be edited or created in the pedestrian configurator (see below).

All data sets are stored in *Movie/3DObjects/Characters*.

##### Source asset (Geometry files)

The source asset is a collection of files (3d model, skeleton, textures, etc.) that define the visual appearance of the pedestrian. This collection can be stored in a folder or packed in an MOBJ file.

All pedestrians that are available with CarMaker are delivered as MOBJ-Packages. You will not be able to edit those files. While it is planned that you will be able to provide your own models, it is currently not possible since only IPG's proprietary file formats are supported.

Source assets are stored in *Movie/3DObjects/Characters/\** .

## Animation files

Those are the files where the actual movement is recorded. They can be shared among several models, or can be specific to a model.

Currently only IPG's proprietary file format is supported.

Animation files that can be shared are stored in *Movie/3DObjects/Characters/\*/Common*. Model-specific animation files are stored in the source asset.

### 7.6.5 Pedestrian configurator

IPGMovie allows you to edit/create a pedestrian data set. The configurator is launched thought the menu *File*.

Of interest so far are mostly the settings *Scale* and *Retroreflexive Texture*.

As the collection of available source assets expands, more settings will be available.

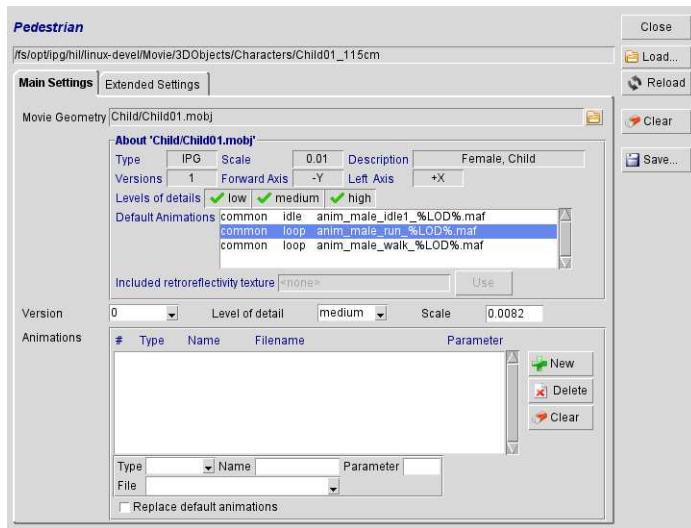


Figure 7.53: Pedestrian Configurator window

## 7.7 Video Data Stream

This feature is available as an optional module. Check *Help > License Details* for more information.

In addition to the output generated to be displayed on the screen and completely independent from that data IPGMovie is able to generate pixel images in a specified format for several virtual cameras placed on the vehicle. This feature is called Video Data Stream (VDS).

The user can define up to 10 virtual cameras which generate data out of a CarMaker simulation. Free parameterizable values are the number of cameras, their position, orientation, viewing angle, resolution and frame rate.

Every virtual camera generates its own data stream which can be exported either to a file or as a video data stream over TCP/IP to be read by arbitrary programs/processes. This export can be used for user specific image processing or as 3-D data provider for an external renderer. In addition, every camera is able to generate a depth map, where the color (greyscale) of each pixel represents the distance between the camera and the displayed object

The interactive views and the VDS frames are synchronised, which means that per frame for the interactive views several VDS frames may be rendered per VDS camera, depending on the respective framerates.

For instance: if the interactive frames are rendered with 60Hz and the virtual camera is parameterized with 120Hz, then typically 2 VDS images will be rendered for each frame for the screen



Figure 7.54: Different outputs of VDS

Factors, that will affect the performance of this feature:

- Graphic card: at each frame, the scene may be rendered several times for each virtual camera, depending on the required framerate. Moreso if fisheye lenses are used.
- System: during the data stream generation, the pixel information is copied into the main memory and needs to be processed by the CPU
- Network: the amount of data to be transferred per network may induce a latency which will cause a time shift between the streamed data and the visualization

Corresponding performance recommendations:

- keep the number of virtual cameras as low as possible
- keep the width and height of the image as small as possible
- keep the frame rate as low as possible
- turn the vertical synchronisation of the graphics adapter off
- keep only one window with one view, and set its resolution as low as possible.
- turn VDS preview off
- keep only the needed rendering effects



Please refer to [section 7.7.6 'Performance considerations'](#) on page 463 for more information and more recommendations.

See the mode of operation and typical configurations below.

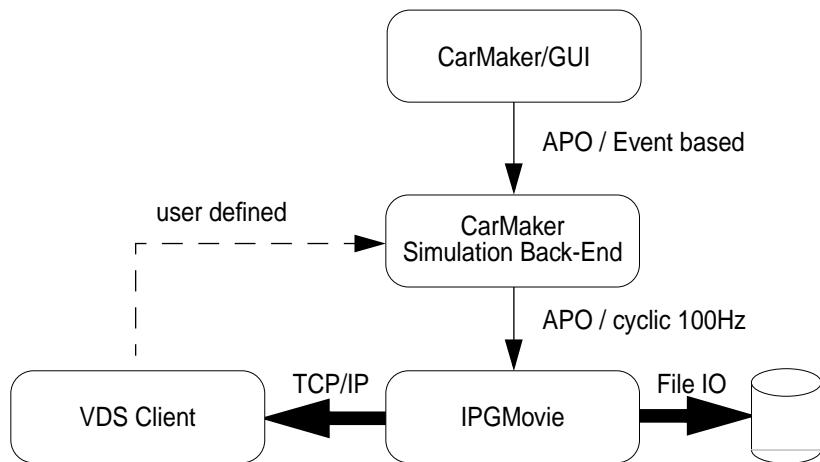


Figure 7.55: Mode of operation

The VDS client can either run as a standalone program or integrated into CarMaker's simulation back-end. When IPGMovie and the VDS client run on the same computer, a maximum transfer rate can be reached (see [section 7.7.6 'Performance considerations'](#)), otherwise the ethernet connection is the bottleneck.

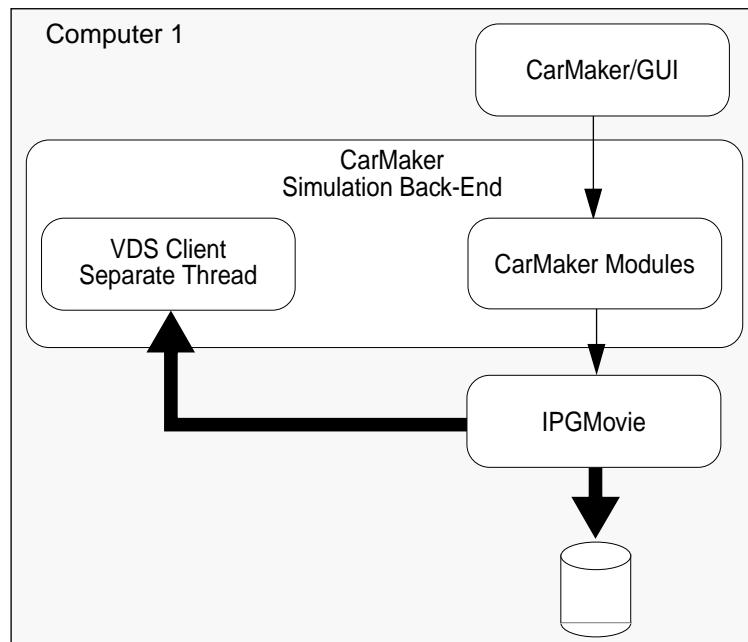


Figure 7.56: VDS client as part of CarMaker simulation back-end

An example for an integrated implementation is provided, see [section 'Client integrated in CarMaker'](#).

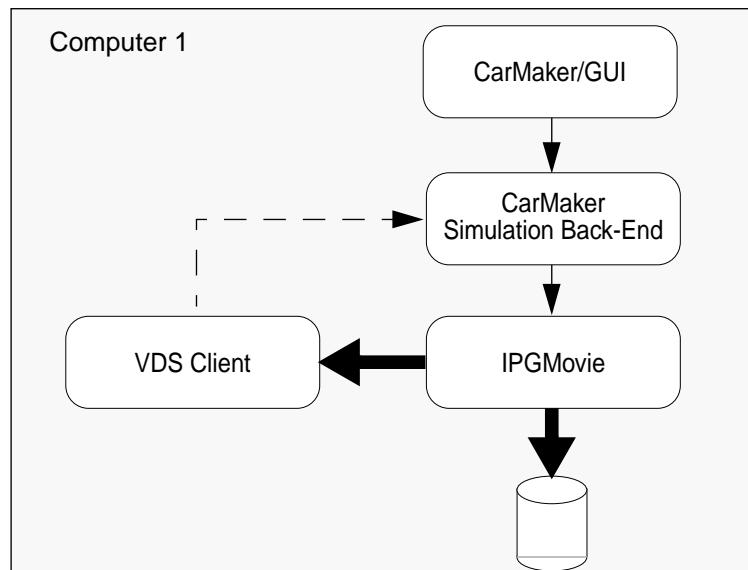


Figure 7.57: VDS client standalone configuration 1

An example for a standalone implementation without any connection between the VDS client and CarMaker's simulation back-end is provided, see [section 'Standalone client'](#).

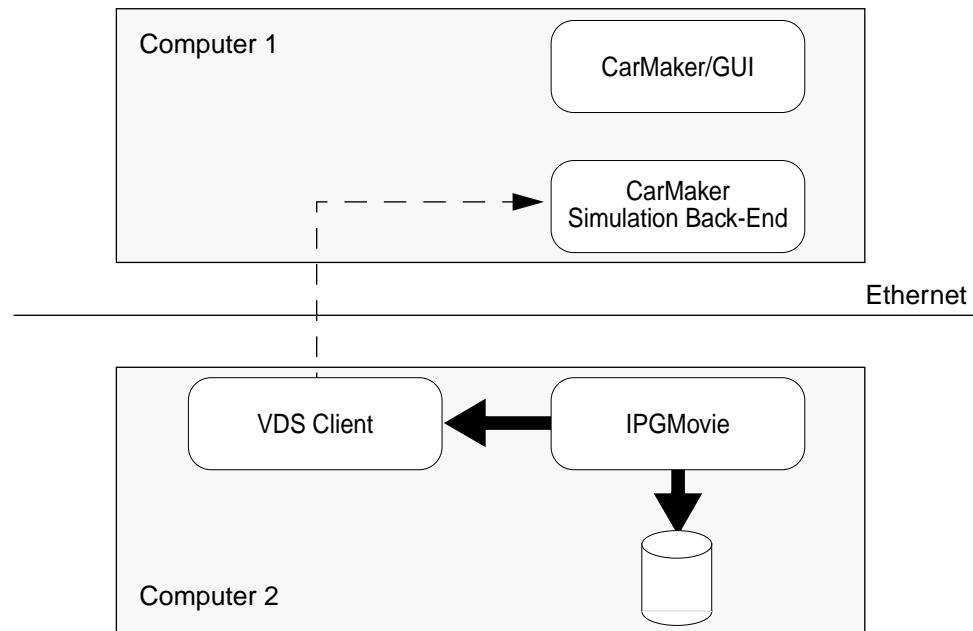


Figure 7.58: VDS client standalone configuration 2

To increase the simulation speed separate the image processing (which is expected as very performance consumptive) from the simulation (Configuration 2). Make sure your Ethernet connection has a big bandwidth, at least higher than the amount of data which needs to be transferred.

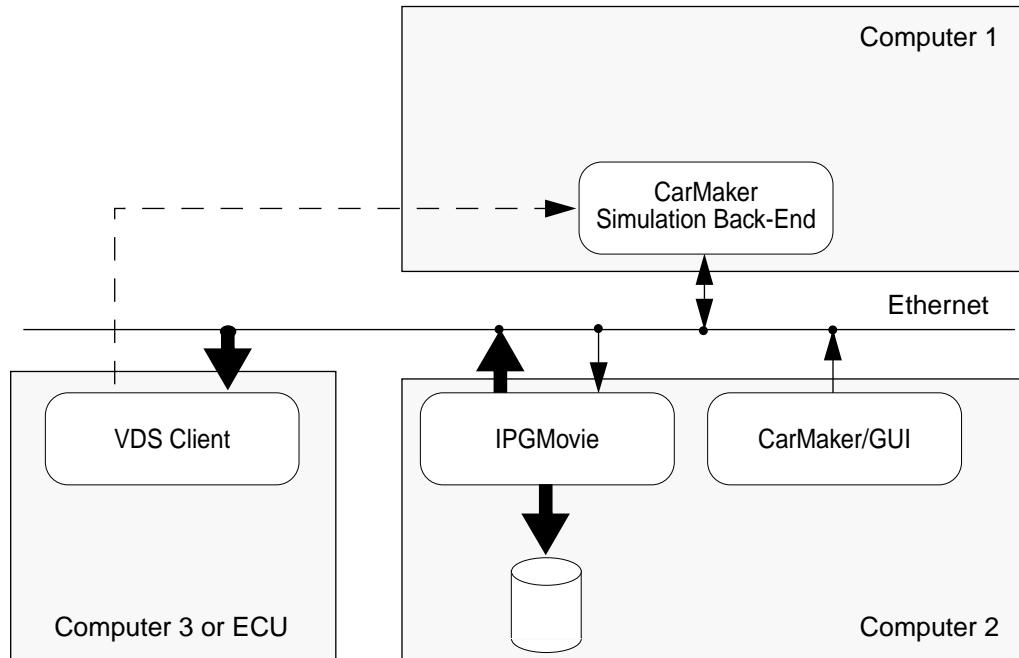


Figure 7.59: VDS client standalone configuration 3

Configuration 3 is typical for realtime applications.

## 7.7.1 VDS Preview

By selecting the respective VDS entry in the *View > Overlay - Top Left* or *Overlay - Top Right* menu, additional windows appear in the upper left and right corner which show the camera view that you have defined before in the *VDS.cfg* file. This is the view which will be exported. Please note that only one camera can be displayed, but images will be created for all cameras you have defined.

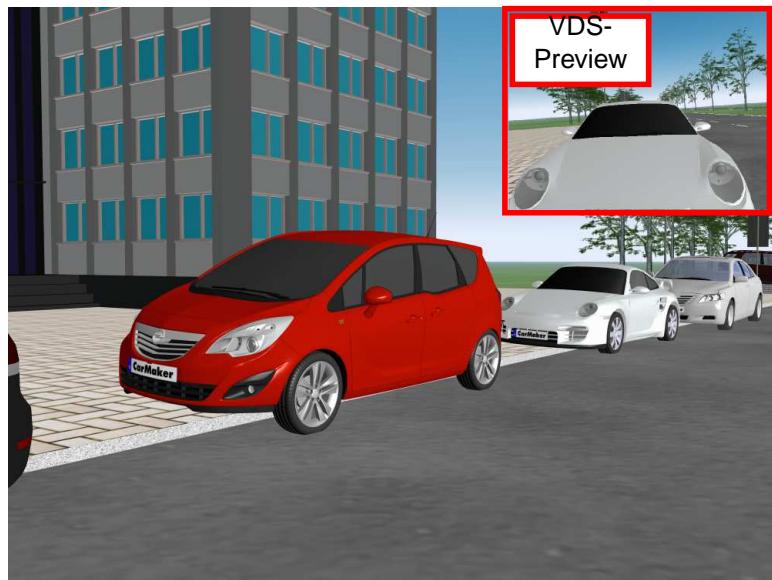


Figure 7.60: Video Data Stream Preview

The preview will be displayed in the mode that was selected for the export (RGB, grey or depth). The VDS preview requires that the scene can be rendered at least twice, therefore the preview should be deactivated for maximum performance.

## 7.7.2 Configuration File *VDS.cfg*

First step to use the Video Data Stream is to set up a configuration file for IPGMovie. Create a new text file and save it as *VDS.cfg* in your *<ProjectDir>/Movie* directory or use the prepared configuration file in *Examples/VDS* and adapted to your needs. There are many parameters you can modify. Some of them are optional, if the values are missing they will be replaced by default values.

<b>Example</b>	" <i>VDS.cfg</i> "
44:	nViews 1
45:	# Camera Settings
46:	0 {
47:	Width       300
48:	Height      200
49:	Export      {grey16}
50:	FrameRate   5
51:	}

This example defines a camera (*nViews=1*) with a resolution of 300x200 pixel which exports 16bit greyscale values with a frame rate of 5 frames/s.

As you can see # is used to place any comments to the file.

The configuration file will be checked every time you start a simulation and if it has been modified it will be memorized again. In order to enforce a new loading of the file, just click on *Reload Geometry*.

The Table 7.10 shows the available options for the global VDS settings that has to be entered at the beginning of the file.

Table 7.10: Global settings

Key	Description	Unit	Default Value
nViews	Number of cameras, respectively defined in a block of the form <n> {...} (see. <a href="#">Listing Example</a> on the last page)	-	0
Socket	Port for the TCP/IP data interface >= 1024. Please select an otherwise unused port, see /etc/services.	-	2210
SyncWindow	Up to which value the simulation program is allowed to hurry ahead of the animation (ignored for HIL). Should be 1.5-2 times higher than the time between two images and minimum 0.03  -1 = no synchronization	s	-1
SyncTimeout	Maximum time the simulation program waits for synchronization. Should be higher then the max. time to proceed with the image processing	s	2.0

Additionally you have to define one or more cameras that stream data into a file or over ethernet by TCP/IP.

A camera is defined by a leading number followed by a data block in curly braces. The numbering starts with zero and can be increased in whole numbers up to nViews-1. The [Table 7.11: Camera/Video channel settings](#) and [Table 7.12: Camera/Video channel settings, specific to fisheye lenses](#) show the parameters that can be modified.

Table 7.11: Camera/Video channel settings

Key	Description	Unit	Default value
Width	Image width	pixel	300
Height	Image height	pixel	200
FieldOfView	Angle of longer image dimension	deg	60 (perspective), 180 (fisheye)
Export {<fmt>}	List of the export formats, details see <a href="#">Table 7.13: Supported export formats</a>	-	{ }
CameraMode	Camera connected to vehicle <i>MoveWithCar, FixedToBodyA (former "SittingIn-Car")</i>	-	FixedTo- BodyA
CameraSmooth	Stiff movement of camera with vehicle ( <i>off</i> ) or (varying) elastical camera movement <i>off, low, medium, high</i>	-	off
Lens	Lens type: direct, fisheye_linear, fisheye_equalarea, fisheye_ortho, fisheye_stereo  See <a href="#">Table 7.12: Camera/Video channel settings, specific to fisheye lenses</a> for parameters specific to fisheye lenses.	-	direct
zrot	Rotation of camera around z-axis in degree ZYX convention	deg	0

Table 7.11: Camera/Video channel settings

Key	Description	Unit	Default value
yrot	Rotation of camera around new y-axis in degree	deg	0
xrot	Rotation of camera around new x-axis in degree	deg	0
VPtOffset_x	Offset of viewing point in x-direction from the origin of vehicle Fr1. If <i>dist</i> value is 0 than this is the mount point of the camera.(see <a href="#">Figure 7.62</a> )	m	3.9
VPtOffset_y	Offset of viewing point in y-direction	m	0.0
VPtOffset_z	Offset of viewing point in z-direction	m	1.1
dist	Distance between camera position and viewing point. Easiest is to define this value as 0.	m	0.0
FrameRate	Frame rate	1/s	10
FrontClip	Distance of front clipping plane. If value < 0.1, bad depth resolution is to be expected.	m	0.1
BackClip	Distance of far clipping plane.	m	4000
FogStart	Starting distance for the fog	m	10
FogEnd	Ending distance for the fog	m	600
FogDensity	Fog density	-	0.05
ShowForces	Tire forces 0 = hide forces 1 = show forces	-	0
ShowSensors	Sensor field 0 = hide sensor field 1 = show sensor field	-	0

The following parameters will have an effect only if the lens is a **fisheye** lens.

Table 7.12: Camera/Video channel settings, specific to fisheye lenses

Key	Description	Unit	Default value
Fit	x, y, diagonal - diameter of image	-	x
Scale	Image scaling	-	1
OffsetX/Y	Principle point in offset X / Y	-	0
DistortionSettings	List of distortion coefficients	-	{1 0 0 0 0}
Vignetting	List of vignetting coefficients	-	{0 0 0 -0.4}
Sensitivity	2x2 matrix of weighting coefficient triples for raw12 and raw12p. Sum must be equal to 1.0!	-	{{0.1 0} {1 0 0} {0 0 1} {0 1 0}} (Bayer filter pattern)

You can choose between several export formats that can be defined by the parameter “Export”. Just replace <fmt> by the values below. You can export multiple formats in only one channel. Just specify all export formats in the curly braces separated by a whitespace.

Table 7.13: Supported export formats

Export format	Description
rgb	RGB, 3byte per pixel, first - red value, second - green value, third - blue value
grey	Greyscale picture, 1byte per pixel
grey16	Greyscale picture, 16bit per pixel
depth	Depth map (distance from the camera), 4byte float per pixel This format has the maximum depth accuracy. The unit is meter.
depth16	Depth map, 16bit integer per pixel, This format has a range from 0 to 655.35 meter. One digit is equivalent to one centimeter.
raw12	Raw formats allow for the generation of image data by combining the red, green and blue components of a color pixel according to the weighting coefficients specified in the Sensitivity parameter list.
raw14	
raw16	The combined RGB values are then coded over 12, 14 or 16bits. 24bits are sent
raw12p	For each pixel in the raw12p format, 24bit of data are sent, encoding two neighbor pixels allowing for a tight packing of the data. <b>WARNING:</b> since this format effectively halves the width, it <i>cannot</i> be combined with any other formats.

#### Example

```
"VDS.cfg"
52:      nViews 2
53:      # First camera
54:      0 {
55:          Width      1280
56:          Height     720
57:          Export     {rgb}
58:          VPtOffset_x 3.1
59:          VPtOffset_y 0
60:          VPtOffset_z 1.1
61:          FrameRate   25
62:      }
63:      # Second camera
64:      1 {
65:          Export     {rgb grey16 depth}
66:          FrameRate   5
67:      }
```

This example defines two cameras (nViews=2) and sets port 2210 for data stream.

- The first camera with the ID “0” exports a 1280x720 RGB-bitmap with a framerate of 25 pictures per second. The view is not rotated but has a displacement in x/y-direction.

- The second camera has a default resolution and exports three different images through one channel.

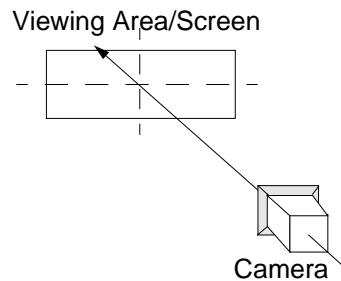


Figure 7.61: Camera is always directed to the center of the screen

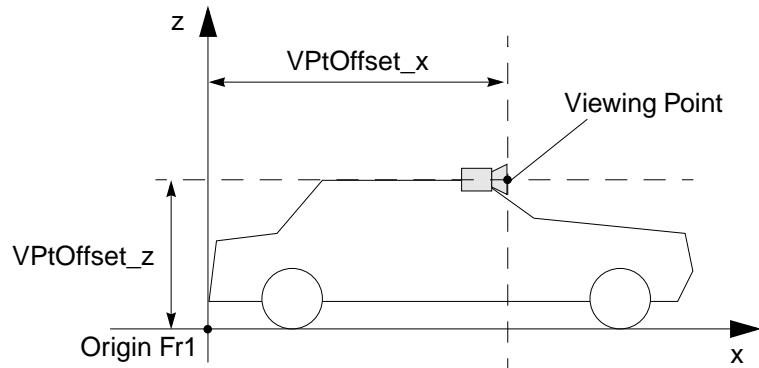


Figure 7.62: *dist* equal to 0, camera and viewing point are identical to 0

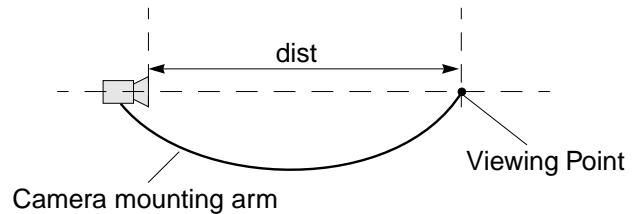


Figure 7.63: Configuration when *dist* not equal to 0

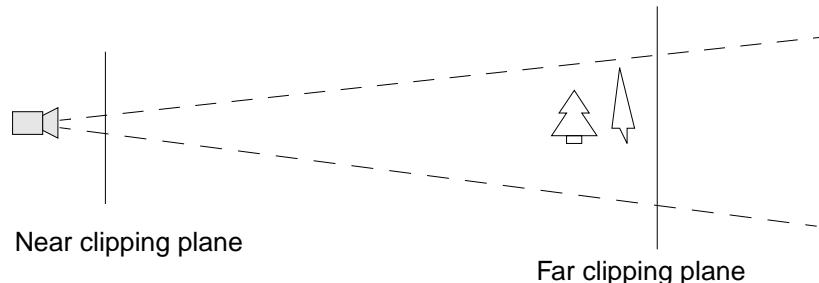


Figure 7.64: Only objects between the clipping planes can be seen

### 7.7.3 Streaming data over TCP/IP

IPGMovie blocks the configured TCP/IP port when using the VDS functionality. No other process, including a new IPGMovie application, can use this port when it is being used by the first IPGMovie application.

After connecting to IPGMovie, a process/program information about the software will be transmitted.

```
/*IPGMovie 5.1.4 1/23/17\n
```

Every transmitted image gets a header information with the following format:

```
/*VDS no fmt time wixhe len\n
```

Table 7.14: Transmitted header

Variable	Type	Description
no	int	Output channel/camera
fmt	char[64]	Exported image format
time	double	Actual simulation time
wi	int	Width of the image in pixel
he	int	Height of the image in pixel
len	int	Length of the image in bytes

In general, header information will begin with \* and will always end with \n. The length is variable (max. 64Bytes including the \n). Then image data will be transmitted. The amount of data depends on the requested image format.

The data export over TCP/IP is similar to the exported file formats (see [section 7.7.5 'Export image data to file'](#)) only the header is different (as described above).

## 7.7.4 Video Data Stream Client

A VideoDataStream client allows you to receive data over TCP/IP and to process image from a connected computer.

Two examples for a VDS client are located in the *Examples/VDS* directory of your installation. Both examples use the same image process algorithm: the calculation of the minimal depth value provided by the depth camera.

The user defined image process algorithm for every single image can be inserted in the *VDS\_GetData()* function. The pixels for every format are discarded line by line from top to down and from left to right. The length of sent information for every pixel depends on the export format used.

### Hints

- Please note that only one user can be connected to the VDS TCP/IP interface. If a second user starts a VDS communication with IPGMovie when another user is still connected, the first connection will be interrupted.
- The VDS client must read out the data quickly otherwise it will block/slow down IPGMovie. This could lead to a less interactive behavior of IPGMovie. Therefore we recommend to export only those images that are really of interest.
- A client can anytime disconnect by closing the used socket.
- Changes in the configuration file does not affect the data stream. Only changes on the TCP/IP port will close the connection to the client.

### Standalone client

This example is to be used as a standalone client for open loop applications. It contains of one single C file “*vds-client-standalone.c*”.

Precompiled versions for Linux 2.6 (*vds-client-standalone*) and Windows (*vds-client-standalone.exe*) are available. The programs must be started from a shell/command window because they produce just text on standard output. After you connect the client with IPG-Movie the actual date and program version are printed. This is then followed by the image information.

```
[hostname-WorkDir] 1) vds-client
VDS: Connected: IPGMovie 3.4.1 2009-8-14
> *VDS 0 depth 0.000 640x480 1228800
> *VDS 0 depth 0.323 640x480 1228800
> *VDS 0 depth 0.414 640x480 1228800
> *VDS 0 depth 0.506 640x480 1228800
> *VDS 0 depth 0.597 640x480 1228800
> *VDS 0 depth 0.697 640x480 1228800
```

#### Usage:

```
vds-client [Options] [MovieHost]
-v          verbose, additional output will be generated
-p          TCP/IP port number, default = 2210
MovieHost host to connect to, default = localhost.
```

To recompile the application use for Linux:

```
gcc vds-client-standalone.c -o vds-client-standalone
```

and for Windows:

```
gcc vds-client-standalone.c -o vds-client-standalone -lws2_32
```

## Client integrated in CarMaker

This example is to be used as a closed loop application, where the image processing is part of CarMaker' simulation back-end. It contains of two files: *vds-client.c*, *vds-client.h*. The image processing generates information which will be immediately used during the simulation to influence the behavior of the vehicle (e.g. lane keeping assistant, sensor data fusion).

To use this application you need to recompile your user accessible code in order to generate a new CarMaker executable. Just copy *vds-client.c* and *vds-client.h* to your *src* or *src\_cm4/* *src\_tm4sl* (depends on the used environment: stand alone simulation environment or Simulink simulation environment) and modify *User.c* and *Makefile* like described bellow. Optional parameters can be set in *SimParameters*. For further information, see chapter "SimParameters" in the Reference Manual.

### User.c

```
#include "vds-client.h"
...
int User_Init (void) {
...
    VDS_Init();
}
int User_TestRun_Start_atEnd (void) {
...
    VDS_Start();
}
```

### Makefile

```
OBJS =
...
vds-client.o
```

After starting the new compiled application three more UAQ's are available.

Table 7.15: New VDS Client UAQ's

Name	Unit	Description
VDS.MinDepth	m	Minimal depth value computed by implemented algorithm
VDS.nBytes	-	Number of received bytes
VDS.nImages	-	Number of received images

### 7.7.5 Export image data to file

In addition to the features described above you have the opportunity to export the VDS images to a file. You can find this feature in the *File > Export Video/Images* menu of IPG-Movie.

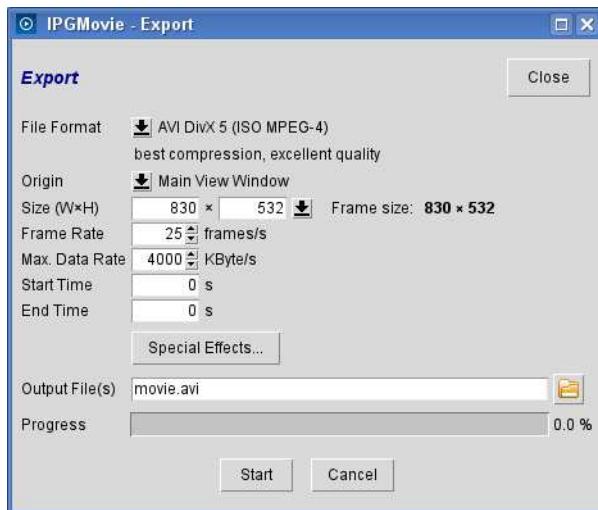


Figure 7.65: IPGMovie - Video / Image Export

You can choose between JPEG, PNG, PPM and PGM greyscale images for greyscale and three different export formats for the depth map.

#### JPEG images

Saves rgb information as a JPEG image.

#### PNG images

Saves rgb information as a PNG image.

#### PPM images (24 bit, true color)

Saves rgb information as a 3byte map.

## PGM Greyscale 8/16bit

Please pay attention: 16bit greyscale pgm files are not supported by many picture viewers. Please make sure your viewer is supporting 16bit images, otherwise 16bit images may look like the following picture.

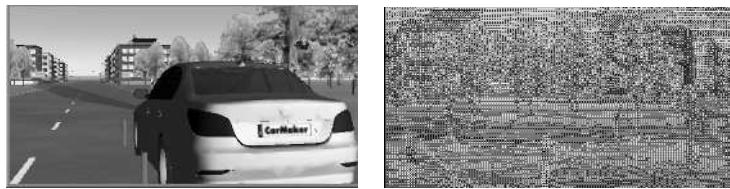


Figure 7.66: 16bit viewer vs. 8bit viewer

## PGM depth image (linear)

Saves depth information as a 16bit greymap. It exports the distance from the camera in a linear scaling order where one digit is equivalent to one centimeter. It has a range from 0 to 655.35 meter at a byte order of MSB.

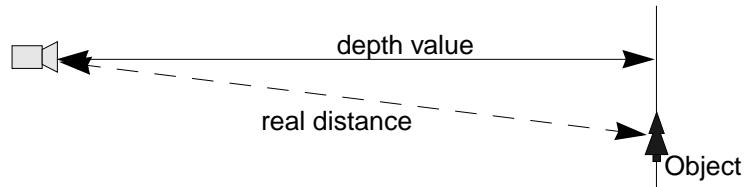


Figure 7.67: Depth vs. real distance

## PGM depth image (linear, float)

The file format is similar to PGM, but data is stored as a 4byte float value instead of a 16bit integer. The advantages lies in the maximum accuracy, you are not limited to 655.35m but you have 2 times more data than in the format before. The byte order is LSB.

## PGM depth image set (image + z buffer linear)

Saves depth information as a 16bit greymap, whereas the pixel values contained match directly the value of the OpenGL Z-Buffer (0x0000 = near clipping plane, 0xffff=far clipping plane). The byte order is MSB.

The course is strongly non-linear and depends very much on near plane and far plane, so it is recommended to use one of the formats above.

Conversion formula:

Conversion OpenGL z to World Z:

$$Z = \frac{Zn \times Zf}{Zf \times (1 - z) + Zn \times z} = \frac{Zn \times Zf}{Zf - z \times (Zf - Zn)} \quad (\text{EQ 6})$$

Z-Buffer resolution:

$$b = \frac{Zn \times Zf}{Zn - Zf} \quad (\text{EQ } 7)$$

$$res = \frac{b}{\frac{b}{Z} - \frac{1}{2^{nbits}}} - Z \quad (\text{EQ } 8)$$

in this case:  $z = 0 \dots 1$  Z-Buffer value is a standardized float value

$$z = \frac{\overline{z_{memory}}}{2^{nbits} - 1} \quad (\text{EQ } 9)$$

$Zn = 0.1$  (near clipping plane, or the value stated in VDS.cfg)  
 $Zf = 4000$  (far clipping plane, or the value stated in VDS.cfg)  
 $nbits = 24$

### General Remarks

The file header is matching the PGM 16bit Standard format. It contains 2 text lines:

```
P5\n
400 300 65535\n
```

The second line contains the width and height of the image and the resolution.  
For the float format, the resolution is replaced by *float*.

```
400 300 float\n
```

These two lines are followed by the image data as binary information.  
The pixels are placed row by row, starting from the top from left to right.

### 7.7.6 Performance considerations

Depending on the amount of data which needs to be transferred by TCP/IP, the simulation speed can be slowed down perceptible. Some calculation examples should demonstrate this behavior.

- A *640x480 rgb, 25Frame/s* camera produces  
 $921600 \text{Bytes}/\text{image} = 900 \text{KB}/\text{image} = 21.97 \text{ MB/s}$
- Two *640x480 grey16bit, 25Frame/s* cameras produces  
 $1228800 \text{Bytes}/\text{image} = 1200 \text{KB}/\text{image} = 29.29 \text{ MB/s}$

This amount of data needs to be transferred from IPGMovie to the VDS client. Current benchmark results show:

- IPGMovie and VDS client run both on the same computer. The transfer is done locally.
  - Maximum transfer rate: 340 MB/s
  - No problems are expected.
- Data transfer done over FastEthernet (100MBit/s), WindowsXP -> WindowsXP or WindowsXP -> Linux.
  - Maximum transfer rate: 3.3 MB/s
  - Very low simulation speed is expected. In worst case, transferred frames may be lost.

- Data transfer done over Fast Ethernet Linux -> Linux (same hardware configuration than above!).
  - Maximum transfer rate: 11.7 MB/s
  - Low simulation speed is expected, lost frames are not expected.
- Data transfer over GigabitEthernet (1000Bit/s)
  - Maximum transfer rate: 80MB/s
  - No problems are expected.

### General performance recommendations

- Turn the vertical synchronisation of the graphics adapter off (the option should be available in the properties of the graphics adapter)
- keep only one window with one view, and set its resolution as low as possible. This will ensure that the interactive views take as less rendering time as possible to delay the VDS rendering as less as possible
- turn VDS preview off - the VDS data is sent as soon as a client is available, the preview is not needed
- try to make sure you use 3D models with a low amount of polygons
- keep only the needed rendering effects. Use only if needed
  - puddles / real-time reflection of the road surface
  - real-time reflections on vehicles
  - active lights
  - fisheye lens
  - specific output formats like raw12/raw14 and so on
  - ...

## Chapter 8

# Test Automation

CarMaker offers different possibilities to make your simulation work a lot easier, for example the test automation. Using it, you perform simulations with varying parameters completely autonomously. There are three ways to implement a test automation:

- Test Manager
- ScriptControl
- External Tools

The three possibilities will be explained in detail in this chapter.

## 8.1 Test Manager

A test series in the Test Manager basically consists of TestRuns that are executed automatically one after another. However, the Test Manager offers a lot of functionalities to optimize the preparation, execution and analysis of your TestRuns: You can use variations to modify parameters of a TestRun instead of creating a TestRun for each scenario. You can add script files to define certain actions which should be executed at the beginning or at the end of each simulation. The definition of criteria can be used to judge the results of a simulation at one glance or even to abort the execution of the test series once a user defined condition is met. To open the Test Manager GUI, select Simulation > Test Manager.

All the TestRuns, variations and scripts that make up a test series in the Test Manager are executed consecutively starting from top to bottom. An overview of the execution order is given in the clear view, marked as box A in the picture below. In box B you can find a description of the current test series (if activated under View) or you can place settings such as select a TestRun or enter a value for a variation. Examples will be given in the following.

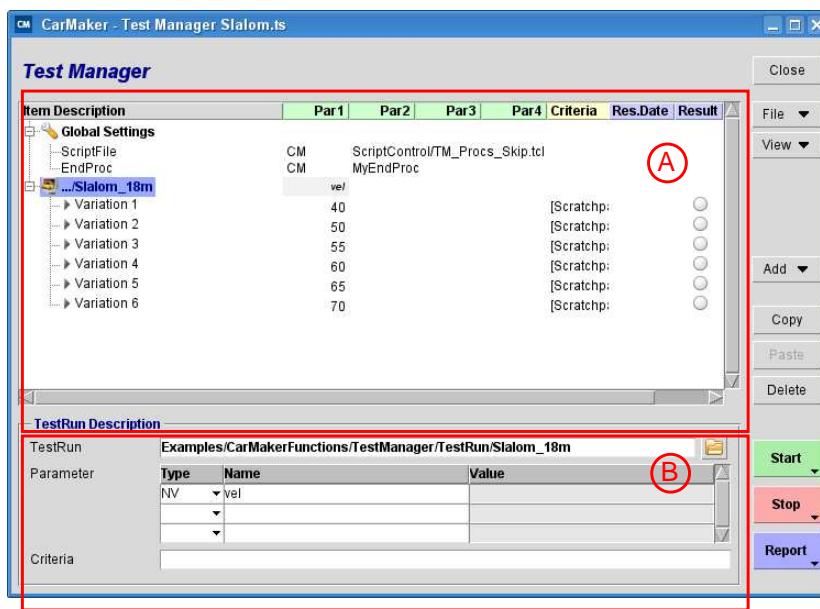


Figure 8.1: Clear view (A) and description window (B) of the Test Manager GUI

## 8.1.1 The Test Manager dialog

In this section the general handling of the Test Manager and the meaning of the main buttons will be explained.

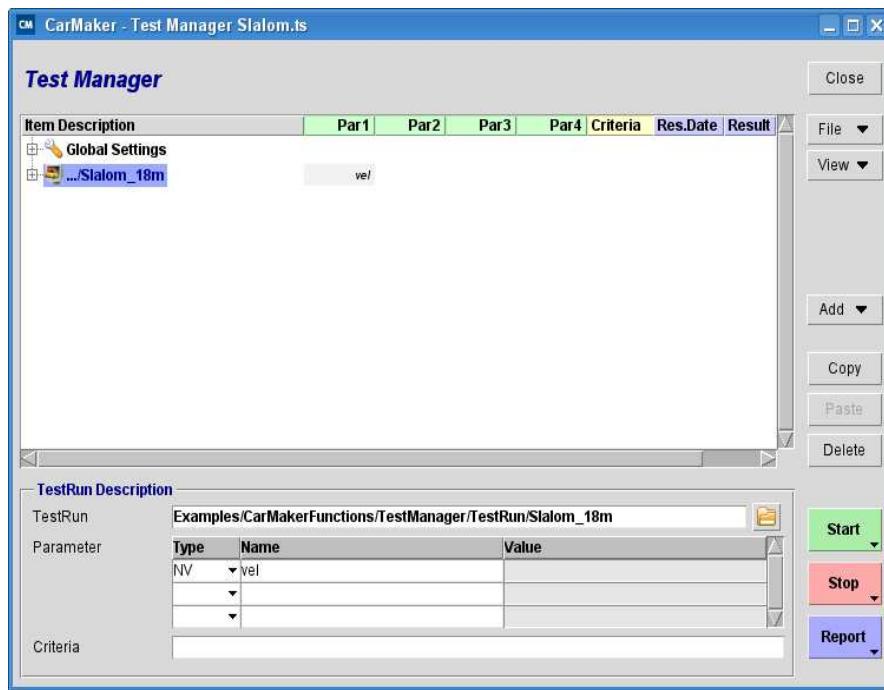


Figure 8.2: The Test Manager dialog

**File** Here you can save your test series or load a predefined configuration. The file you generate contains all settings you made to configure your test series. To differentiate between a TestRun and a test series, the corresponding Infofiles have the extension .ts.



Please note: In the File menu you cannot load a TestRun or save changes to a TestRun, but only entire test series. When you select a Test Manager script instead of a TestRun in the CarMaker GUI (File > Open), the Test Manager pops up automatically and loads the selected test series.

**View** The View button offers the possibility to change some view settings. You can select *TestSpace* and a new window pops up that gives you an overview of all *TestSpace Variables* used. Moreover, the column *Result Dates* can be deactivated.

**Add** The Add button can be used to load different elements of a test series as TestRuns and variations which should be executed consecutively. The elements can be combined to a group, settings that apply for the whole group can be inserted as well as tcl scripts which you know from ScriptControl. Please find further information in [section 8.1.2 'Creating a Test Series'](#).

**Delete** This button serves to delete items from the execution list.

**Start** This button lets you start the execution of your test series. The triangle indicates additional options that appear by keeping the button pressed. E.g. only the selected parts of the test series can be simulated.

**Stop** With this button you can abort the simulation.

**Report** This feature enables you to create a report which summarizes the current test series and its simulation results. Keeping the button pressed lets you configure your own report template.

The context menu which is accessible via right-click provides some additional features which may be useful when working with the Test Manager extensively:

- Review** This feature allows a run through the whole Test Series without performing the actual simulation (i.e. the simulation program will not be started). This can be useful to verify the existence of all parameter files needed for a later simulation and to check the accompanying Tcl scripts for possible errors. During a review all Tcl procedures are called as they would be called during a normal run, characteristic values are initialized with their start values and criteria are evaluated on that base. Since no results are available diagrams will not be plotted.
- Shuffle** When performing a bunch of simulations it is possible that hidden dependencies between subsequent simulations influence the overall outcome. With this shuffle mechanism the pre-defined order of simulations can be changed randomly without breaking the logical order within the Test Series.

## 8.1.2 Creating a Test Series

### Groups

To create a well structured test series it is recommendable to gather related tests in groups. Apart from the clearer arrangement, groups have another advantage: settings can be defined that apply to all TestRuns and variations within that level. This feature is quite convenient if you would like to set the same parameters for a number of tests since you can avoid declaring the same settings in each variation block again (see *Global Config*).

To insert a group go to the *Add* button and select *Group*. In the settings window you can then enter a name for your group.

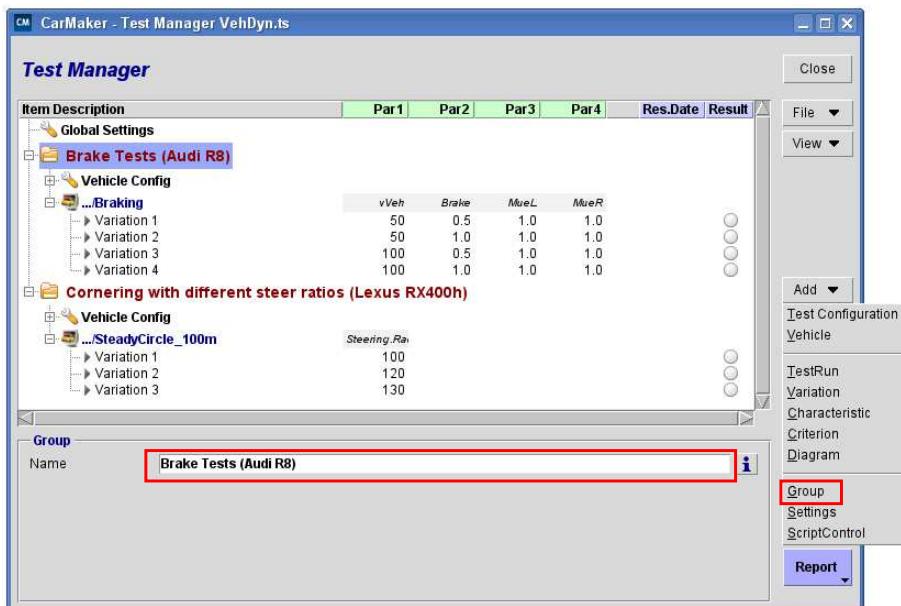


Figure 8.3: A test series with different groups

### Test Configuration

The Test Configuration item is designed as a link to the Test Configurator wizard which can be used to create large test series based on predefined test catalogs, so called TestWare Packages. Adding such an item to a test series allows the user to define the content of the

items subtree by making use of the Test Configurator's functionality. For more information regarding the Test Configurator, have a look at [section 8.1.4 'CarMaker Test Configurator' on page 480](#).

The Test Configurator is accessible by double-clicking on the test configuration item in the treeview or by clicking the "Edit" button on the right side of the detailed view.

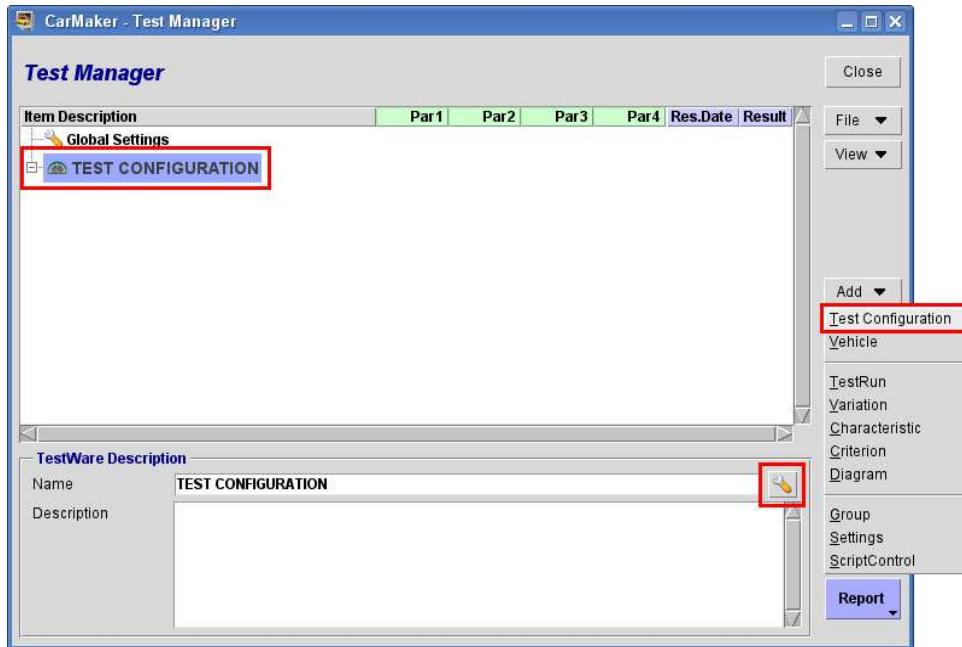


Figure 8.4: Access to the TestWare wizzard through the test configuration item

## Vehicle Configuration

The vehicle configuration item is a convenient way for changing basic settings of your vehicle without making use of a dedicated settings block.

On the *Configuration* tab of the vehicle configuration item's detailed view the user can choose the test vehicle which will be used for the following simulations, configure its tires and trailer, as well as define the arrangement of different loads on the vehicle. All these tasks are handled by the familiar CarMaker file dialog for convenient selection of the desired data sets.

Besides that user-defined Key Values (KValues) and Named Values (NValues) can be specified on the *Parameters* tab for changing certain properties of the chosen vehicle. For a better understanding of the concept behind NValues and KValues please refer to [section 8.3 'Variable Types'](#).

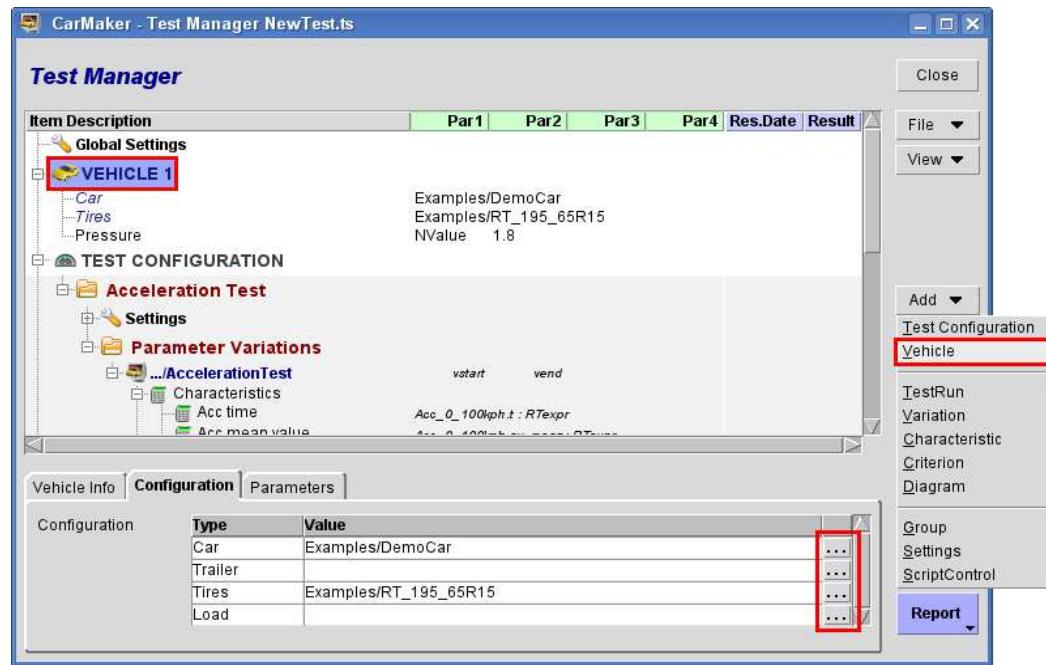


Figure 8.5: Utilizing the vehicle configuration item

## Characteristic Values

When performing a simulation it often is interesting to calculate some characteristic values which help to rate the outcome of the simulation at the end. The characteristic value item helps to define such characteristic values and specify the calculation rule they are based on.

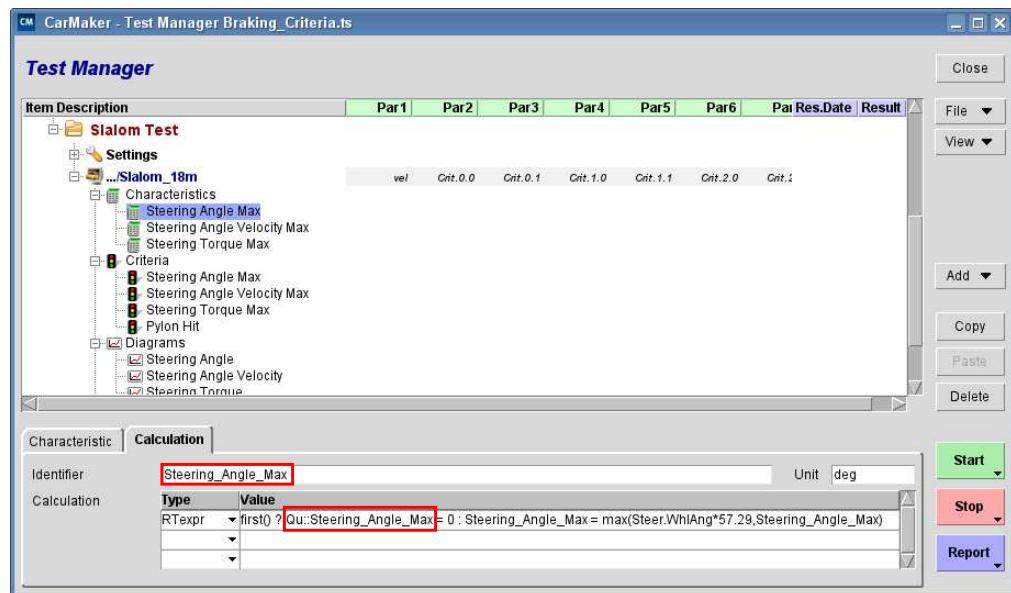
For each characteristic value you have to define a unique identifier which can later be used in a criterion or diagram as reference name (see [Figure 8.6](#)).

Basically there are two different approaches for calculating a characteristic value in Car-Maker, the online and the offline calculation.

### Online calculation

The value is calculated for every time step during the whole simulation by applying a user-defined Realtime Expression (for details on Realtime Expressions refer to [section 'Real-time Expressions'](#) in the appendix). This is applicable to those characteristic values whose calculation only depends on previous time steps.

For calculating a characteristic value online choose *RTexpr* as calculation type and specify your Realtime Expression in the designated entry field (see [Figure 8.6](#)). The Realtime Expression must define a new quantity whose name is identical to the previously specified identifier of the characteristic value.



[Figure 8.6: Defining a characteristic value with Realtime Expressions. Identifier and the new Quantity must have the same name.](#)

### Offline calculation

The value is calculated after the simulation has finished in a post-processing step by analyzing the stored result data. This is basically applicable to all characteristic values.

For calculating a characteristic value offline the user has to specify a Tcl script which provides functions for calculating the desired value. The path to this script file must be declared in a *Settings* block (see [section 'Settings'](#)) prior to the definition of the characteristic value. Additionally, the user has to specify the name of the function which performs the actual calculation.

This function is automatically called by the Test Manager after the actual simulation has finished. Choose *EndProc* as calculation type and specify the function name in the designated entry field. The Tcl function that you specify has to match the following prototype:

```
proc MyEndProc {key args} {
    # Perform calculation
    set value ...
    TestMgr::SetCharacteristicValue $value
}
```

The last command in this function is mandatory and assigns the calculated value to the characteristic value in the Test Manager workspace.

In some cases it is essential to perform some initialization steps before the actual simulation starts (e.g. declaration of variables or activation of the data storage process). For this choose *StartProc* and specify the function name in the designated entry field. Again the Tcl function that you specify has to match the following prototype:

```
proc MyStartProc {key args} {
    # Initialize variables
    set value ...
}
```

## Criteria

For the evaluation of a completed simulation it is often necessary to define one or more criteria which compare the simulation results with given reference values. This is possible by adding one or more criterion items to the test series. The result of the evaluation of such a criterion can either be *good*, *warn* or *bad*. To distinguish between these results different constraints can be specified in the criterion item's detailed view (see Figure 8.7).

The constraints are written as Tcl expressions and must evaluate to either true or false. Within the Tcl expression you can call an arbitrary Tcl function, access the current value of a quantity or retrieve the value of a previously calculated characteristic value (both via the internal CarMaker Tcl command *get*).

These criteria also control the overall result of the simulation in a way that the worst individual result defines the test result.

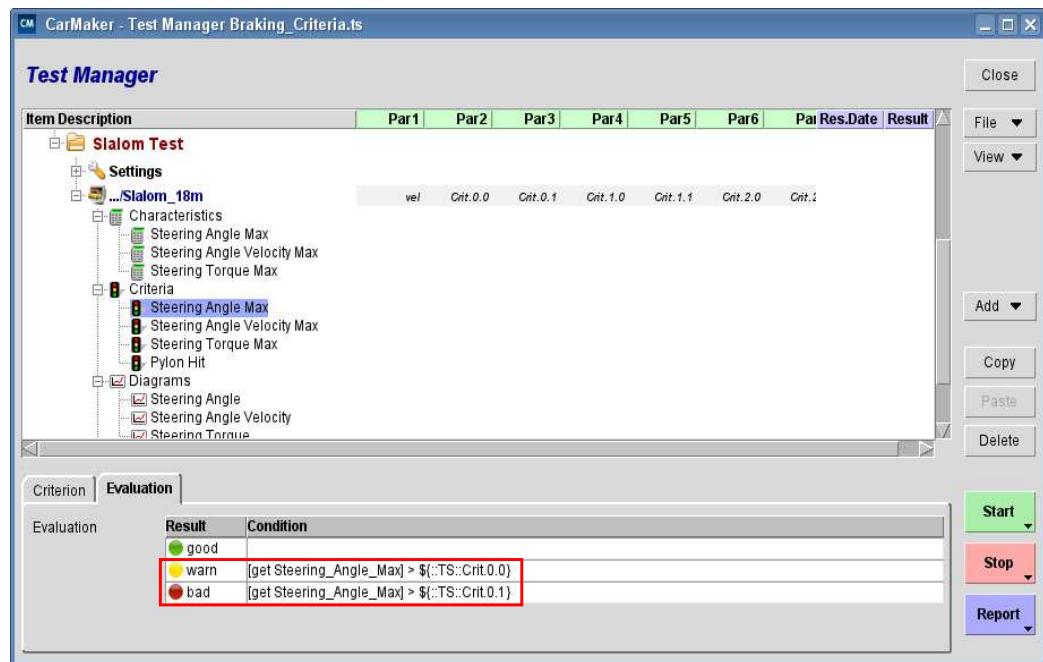


Figure 8.7: Criterion definition

## Diagrams

A new Test Manager feature is the possibility to define diagrams in a test series. These diagrams are plotted after the simulation has come to an end and can then e.g. easily be integrated into CarMaker's report output for documentation purposes.

There are three different diagram modes which are supported:

- quantity(s) vs. time
- quantity(s) vs. quantity
- characteristic value vs. variation

It is possible to plot one diagram for each variation or to summarize all variations in a single diagram for a better overview.

Generally the user can choose between three different diagram types:

- line diagram
- point diagram
- vertical bar diagram (characteristic value vs. variation only).

All these settings can be made on the first tab of the diagram item's detailed view (see Figure 8.8).

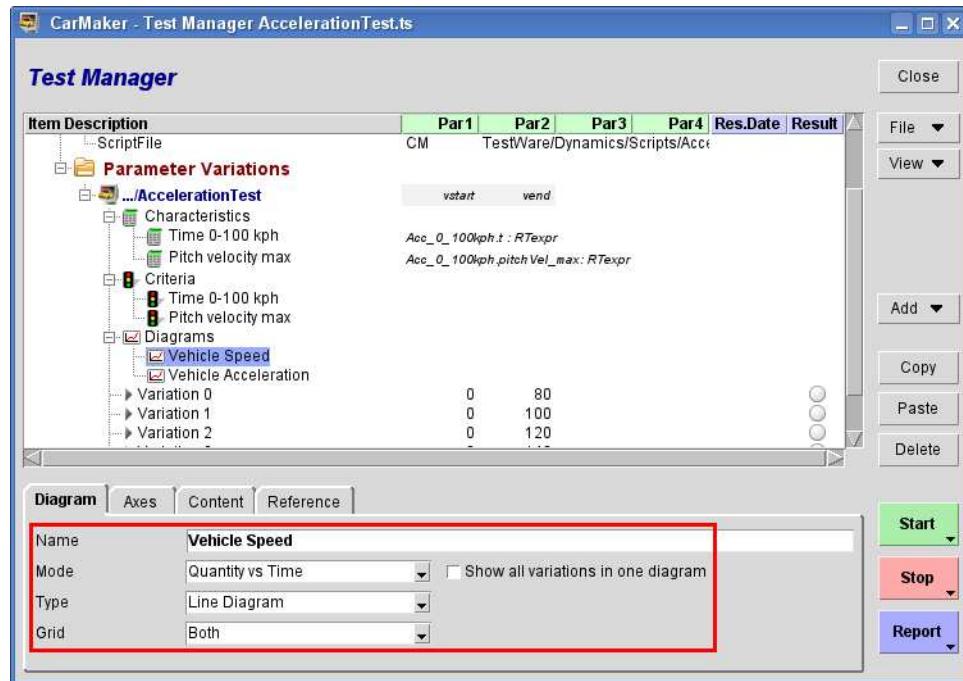


Figure 8.8: General diagram settings

On the next tab the axis properties of the diagram are defined. If two quantities with different units are part of the same diagram it is possible to show two different Y axis in one diagram. Furthermore the user can specify the range of both X and Y axis manually or let CarMaker choose an appropriate range where all values fit in. The additional axis label is printed in the resulting diagram only if more than one quantity is shown on the same axis. In all other cases the quantity's label is shown as axis label (see Figure 8.9).

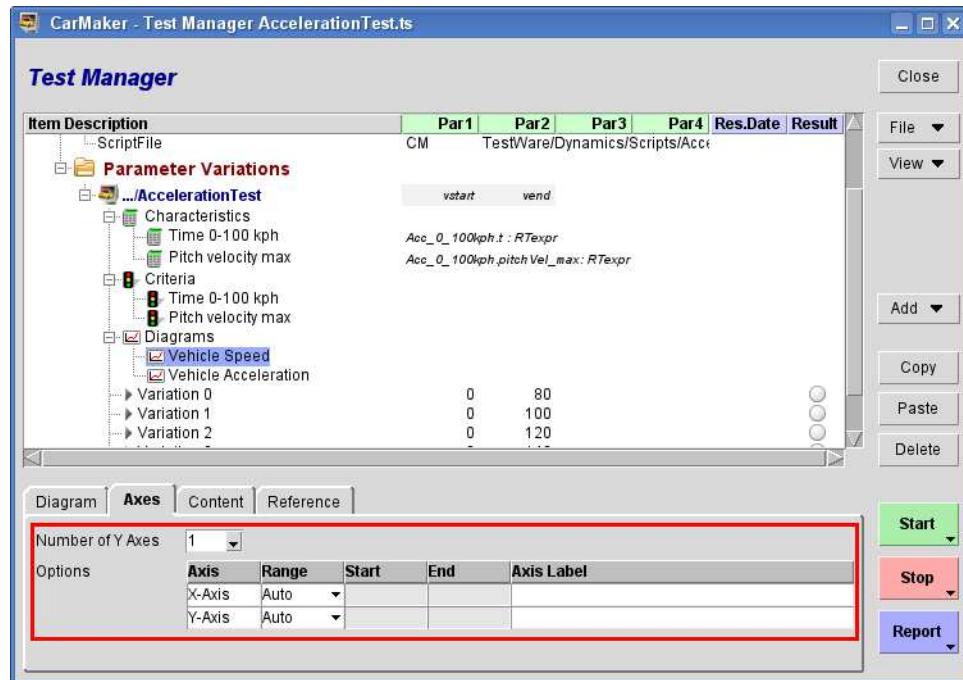


Figure 8.9: Axis properties

On the following tab the actual content of the diagram is defined. The user can select the desired quantities or characteristic values he wishes to display. The values can be converted to other units than SI units by applying a factor and/or offset to them. Again, the label is the text which will be shown besides the axis (if only one quantity is selected) or within the legend (for more than one quantity).

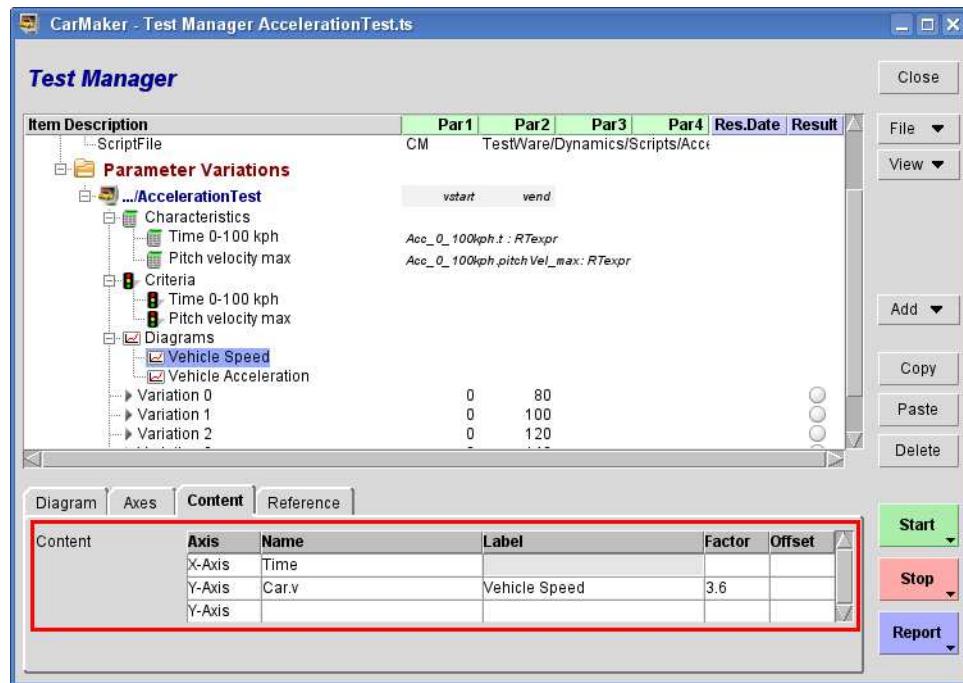


Figure 8.10: Defining diagram content

Finally CarMaker also allows to show reference data within the created diagram. Supported is data from CarMaker erg-Files or CSV data. As stated above unit conversion and curve alignment is possible by using the offset and factor entry fields.

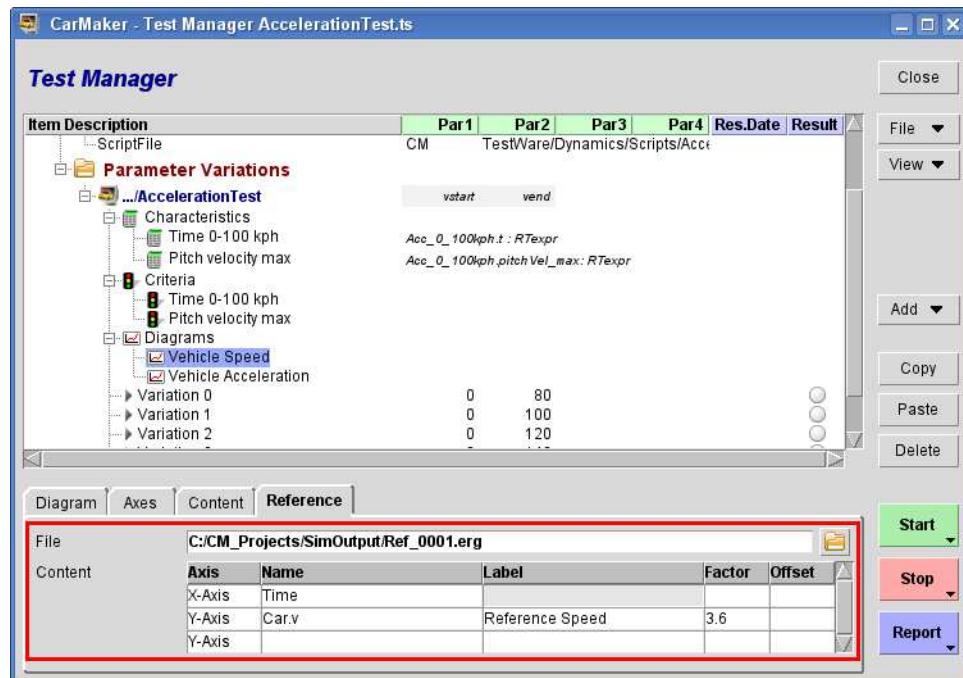


Figure 8.11: Diagram reference data

When generating a report to summarize the simulation results the user can specify whether the generated diagrams should be part of the report document or not. Besides the size of the embedded diagrams can be modified. These changes can be made in the *Report Configuration* dialog which is accessible via a long left-mouse click on the *Report* button in the lower right corner of the Test Manager window.

## Settings

The settings give the user the possibility to make some configurations that are applied not just for one test but at larger scale. E.g. parameter variations can be set which are valid for more than just one variation but a whole group of TestRuns for instance. As a result it is very important, at which position the settings are placed within the test series, as they apply to everything that follows at the same or lower level.

The Test Manager provides two kinds of settings: Global Settings and Settings blocks.

The section *Global Settings* is the start of any test series. Here, you can define settings that apply for everything below: each group, each TestRun and each variation. *Global Settings* can contain script files with start and end procedures (*StartProc*, *EndProc*), TestSpace variables that are globally accessible as well as Named and Key Values.

Contrary to the Global Settings at the beginning, the *Settings* blocks inserted by *Add > Settings* do not apply to the whole test series. They are valid only at the very level they are put in. Inserted at the beginning of a group the settings count for every TestRun and every variation listed in that group. Then, the *Settings* apply for all variations that belong to that TestRun.



There is one exception to this rule: If a *Settings* item is used to define a *ScriptFile*, *StartProc* or *EndProc*, this selection is valid not only for the corresponding group, but for all items (TestRuns, variations, ScriptControl ...) until the selection is overwritten by another *Settings* item. To deactivate start respectively end procs, you can add another *Settings* bloc and set *StartProc* respectively *EndProc* leaving the file *Value* empty.

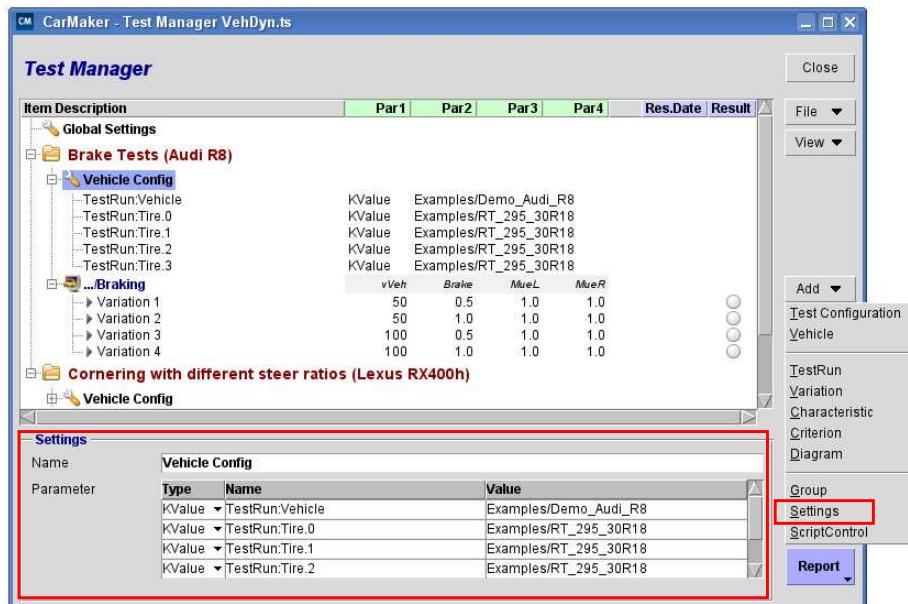


Figure 8.12: Adding a settings block to a group

## TestRun

The next step is to select a TestRun which will be simulated - optionally numerous variations of this test can follow. However, the Test Manager can also be used to execute a row of TestRuns one after the other without any parameter substitutions.

The TestRun has the same functionality as in the plain CarMaker environment: the parameterization of the virtual vehicle environment including a vehicle, tire sets, a track where the test is performed at, a maneuver and the driver who carries out this maneuver. The idea of a TestRun used in the Test Manager is to formulate it as generic as possible including variables to cover as many tests as possible at the basis of the same TestRun.

To insert a predefined TestRun select Add > TestRun.

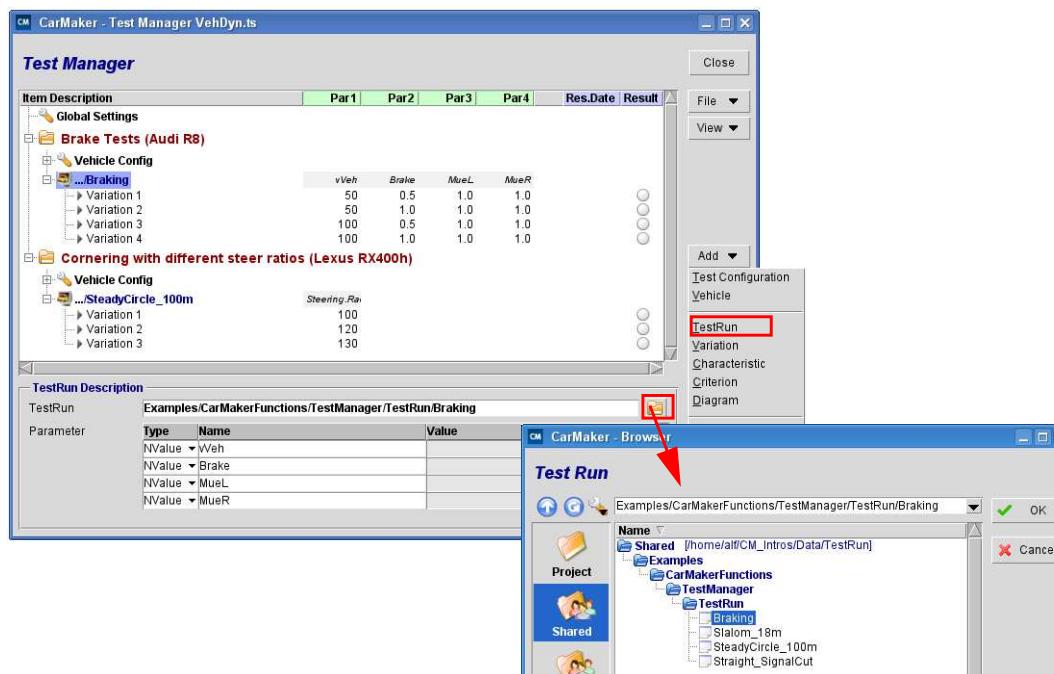


Figure 8.13: Adding a TestRun in the Test Manager dialog

The view in the lower section of the TestRun GUI changes and gives you the possibility to select a predefined TestRun. A click on the folder button opens a new dialog window. It is the same as the one you know from the CarMaker main GUI when you go to File > Open to load a new TestRun. You can load various TestRuns one after the other. This is how you can create a query of tests that are executed automatically in a row by hitting the start button in the Test Manager. You do not need to open each single TestRun manually and start the simulation, this is all done by the Test Manager.

Once the TestRun is selected you can enter the type and name of the parameter variations you would like to perform. Available types are: Named Values, Keys Values and TestSpace variables. To assign values to the variables for each run a *Variation* needs to be added.

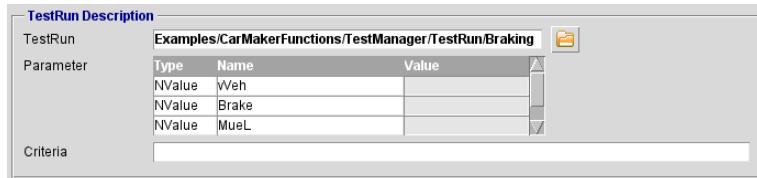


Figure 8.14: Defining variables on TestRun level

## Variations

The autonomous execution of a series of different TestRuns is not the only feature offered by the Test Manager. There is no more need to create a TestRun for each single parameter variation. You can define a TestRun once and enter a variable for the parameter you want to vary. Then, in the Test Manager numerous variations can be added to this TestRun. These variations use all the same TestRun but assign different values to the user defined variables. By defining the variable list on TestRun level (see [Figure 8.14](#)), the list is automatically displayed for each variation added. Available variable types for variations are: Named Values, Keys Values and TestSpace variables (see [section 8.3 'Variable Types'](#)).

To add a variation to a TestRun proceed as follows: after you loaded a TestRun hit the *Add* button once again and select *Variation*.

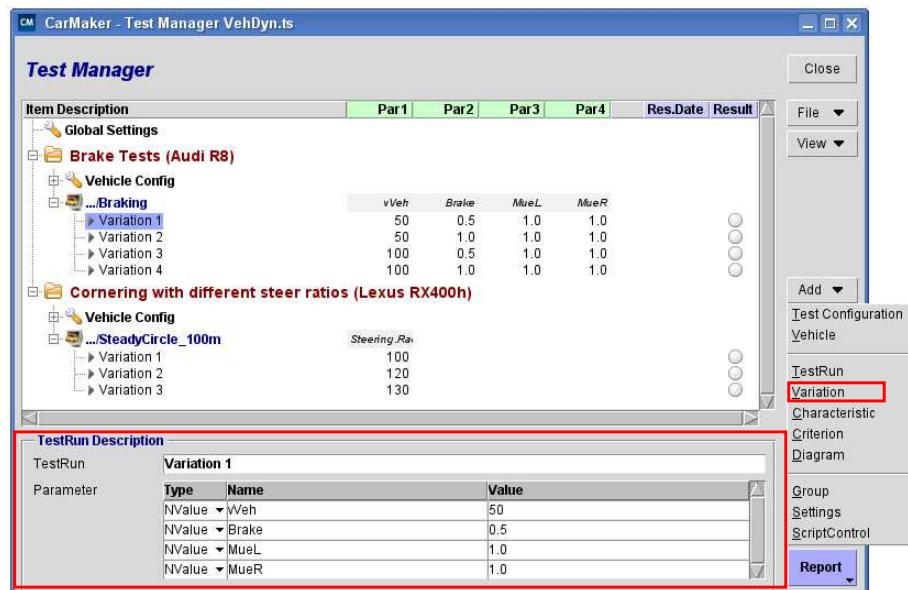


Figure 8.15: Defining a variation

## ScriptControl

This option helps to place ScriptControl commands during the execution of the test series. For this go to *Add > ScriptControl*. Please note, that in the Test Manager native Tcl/Tk code is used and all ScriptControl commands are supported. Please find an overview on ScriptControl in the Programmer's Guide, section 'ScriptControl'.

These scripts can be used to interact with the GUI or to interface external tools like test stand control or calibration tools. Possible applications would be:

- automated generation of AVI files
- firmware download to the ECU
- creation/download of a new model
- execution of Matlab scripts

- access of diagnostic tools

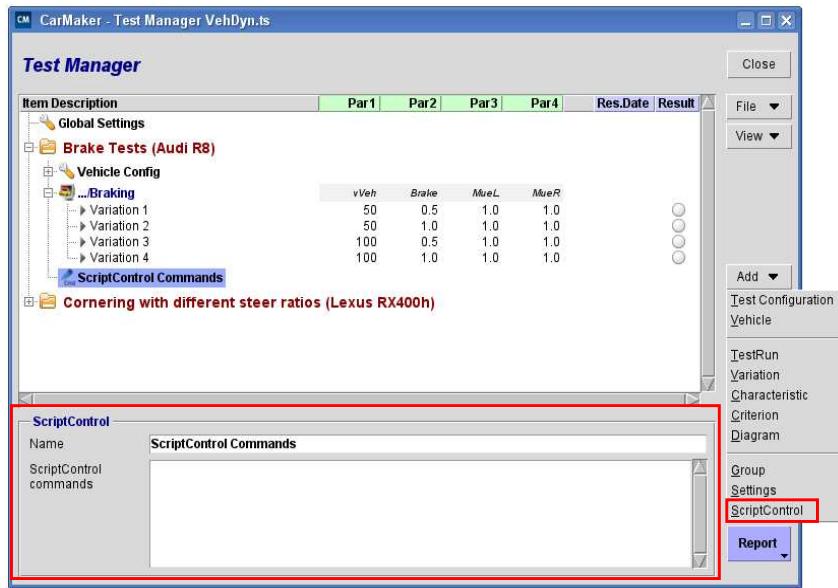


Figure 8.16: Inserting tcl commands directly to the Test Manager

A second possibility to load user defined procedures based on Tcl/Tk is the *ScriptFile* variable type. New functions can be declared in an external script file and need to be called only during the test series. Please find further information in [section 8.3.4 'ScriptFile' on page 495](#).

### 8.1.3 Result Quantification



The following paragraph applies to test series created with older CarMaker versions prior to 5.0. Starting with CarMaker 5.0 the presented concept has been replaced by the dedicated criteria item which is explained in detail in [section 'Criteria' on page 479](#).

Apart from the merely execution of TestRun loops including parameter variations, the Test Manager also provides an interface to quantify the simulations directly. In the result array of the Test Manager clear view lamps can be found which indicate at one glance if a test was successful or not.

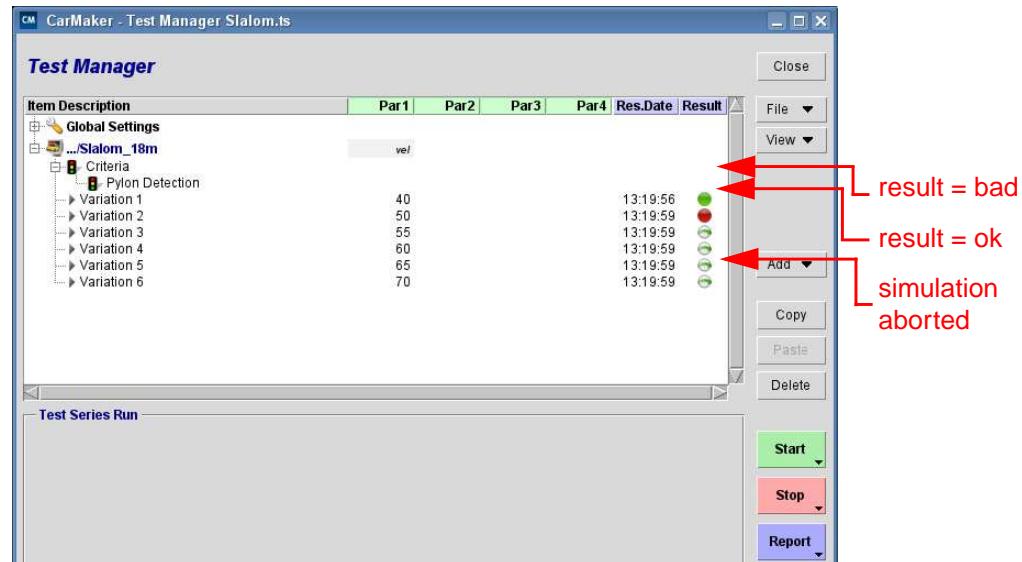


Figure 8.17: Different simulation statuses

The lamps return a so-called *simulation status* which gives the following feedback:

Table 8.1: Simulation statuses available

Icon	Status	Meaning
white	default	no reply from the simulation, not yet executed
red flash	err	simulation could not be executed, failure during evaluation
green	ok	simulation done, result OK
yellow	warn	simulation done, result dubiously
red	bad	simulation done, result not ok (e.g. criteria not matched)
green arrow	skip	simulation not executed but skipped

The status itself can be influenced by:

- using the following function in an end procedure (see [page 495](#)):  
`::TestMgr::SetResult Status`  
 where *Status* can be any of the kinds listed in [Table 8.2: Available variable types in the Test Manager](#)
- the Realtime Expression command `TestLog()` (see [section C.2 'Operators and Functions' on page 603](#))
- the Test Manager [Criteria](#) function

## Criteria

The criterion is an input argument which will mainly be used to evaluate the results of a single test. The value of the criterion is stored to the tcl variable `TestMgr::Criteria`.

There are two ways to implement one or more criteria:

- direct evaluation of the criterion string: see [section 'Criteria' on page 472](#).
- indirect evaluation by calling a user defined evaluation function which sets the simulation status.

For the second approach, a user defined evaluation function needs to be defined in an external script using the Tcl/Tk programming language. The user function can be declared in the same script as the start and end procedures and loaded using the CM-ScriptFile variable (see [page 495](#)). In the criteria field, the user evaluation function can be called using square brackets.

**Example** User defined evaluation function, saved to an external \*.tcl file:  

```
proc EvalMe {val} {
    Log $val
    ::TestMgr::SetResult bad
}
Function call in the criteria field:
[EvalMe 10]
```

Please note, that this approach also requires the usage of the end procedure extensions listed on the previous page.

## 8.1.4 CarMaker Test Configurator

As mentioned above the TestManager's *Test Configuration* item is the link to the new Test Configurator. This wizard allows the user to configure and parametrize pre-defined tests which are part of different so-called *TestWare packages*. Typically a *TestWare package* is a catalogue of several tests which share a common aspect. Those tests are presented in the Test Configurator on a more abstract level than the *TestManager* provides, hiding all implementation-specific details and showing only the essential information to the user. The main advantage of the *TestWare* concept is that one can distinguish between the small group of persons who build the tests and the large group of end users who just use these predefined tests and who do not need to have detailed knowledge about how the tests are build up.

The Test Configuration wizard is divided mainly into three areas as shown in [Figure 8.18](#):

- On the top the user selects the *TestWare package* from which the pre-defined tests should be loaded.
- The main window shows all available tests in the current *TestWare package*, their parameters and all defined criteria. A test is selected if at least one of its criteria is selected.
- At the bottom of the wizard is a frame which displays detailed information of the currently selected item above, e.g. the list of parameters and their current values.

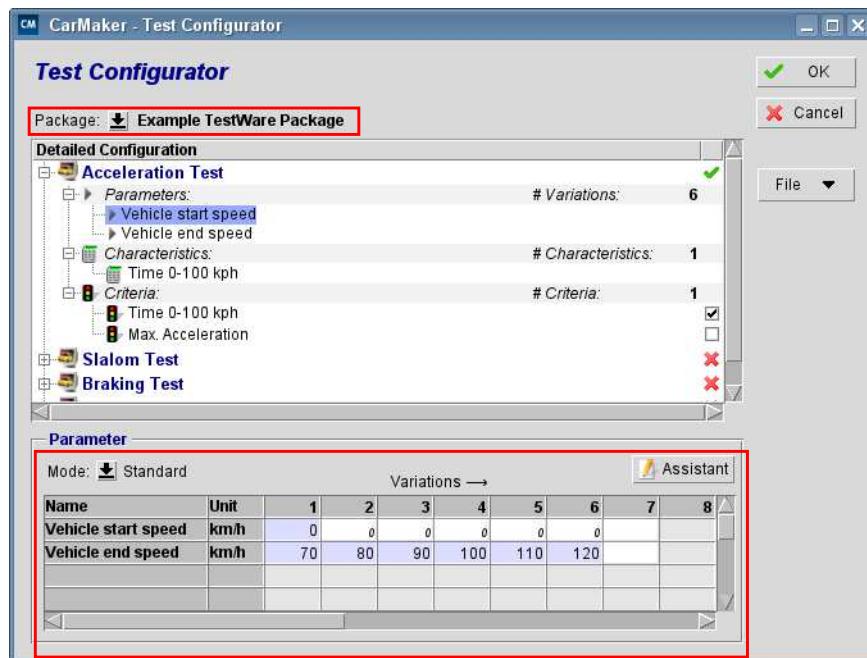


Figure 8.18: Defining parameter variations with the Test Configurator

## Using the Test Configurator

### Defining Parameter Variations

Parameter variations of a test can be created by modifying the available parameter table. There are two different modes for filling in the table:

- In *Standard* mode each column of the table represents a single variation with its own parameter set. For complex tasks a parameter assistant is available which can be used for creating large consecutive ranges of parameter values in a row.

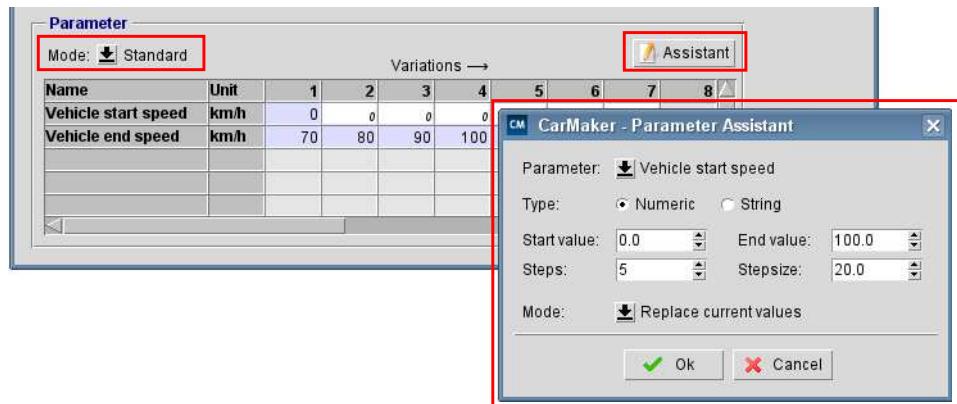


Figure 8.19: Standard parameter mode

- In *Combinatorial* mode the resulting set of variations is formed by combining all values of the different parameters with each other. This quickly leads to a very high number of variations. To reduce this number it is possible to exclude certain variations whose parameter sets hold nonreasonable combinations by applying one or more exclusion filter rules. Such filter rules are formed by boolean expressions and can be defined in the available *Exclusion Filter* dialog.

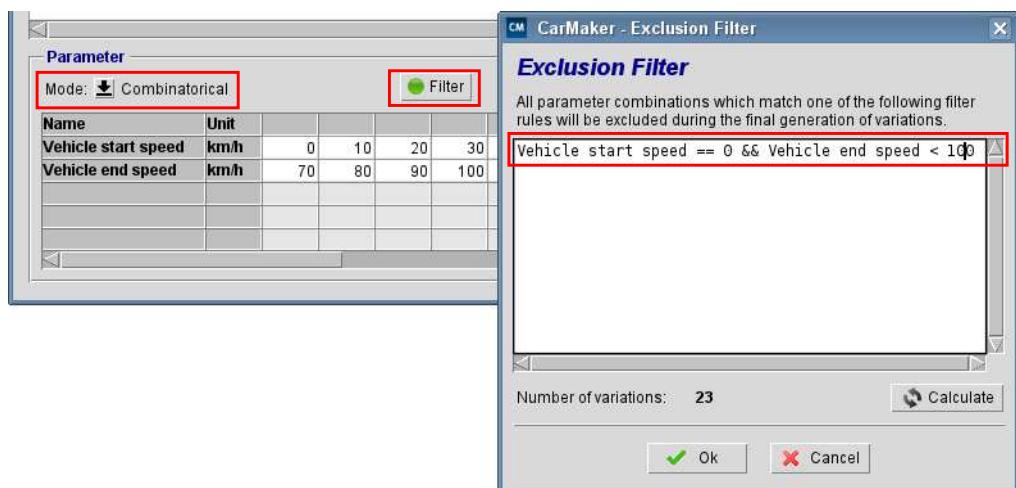


Figure 8.20: Combinatorial parameter mode

### Adjusting the Test Selection

A test within a *TestWare package* is chosen by selecting one or more of its accompanying criteria. Later, these selected criteria will decide over the outcome of the performed simulation. Sometimes it may be necessary to adapt the criteria evaluation boundaries for one or more variations. This can be done by clicking on the desired criterion and change the respective values in the criterion's parameter table. After making the appropriate settings and adjusting everything to the needs it is advisable to save the overall test configuration for later use. Loading and saving a test configuration can be achieved by utilizing the *File* button's drop-down menu on the right side.

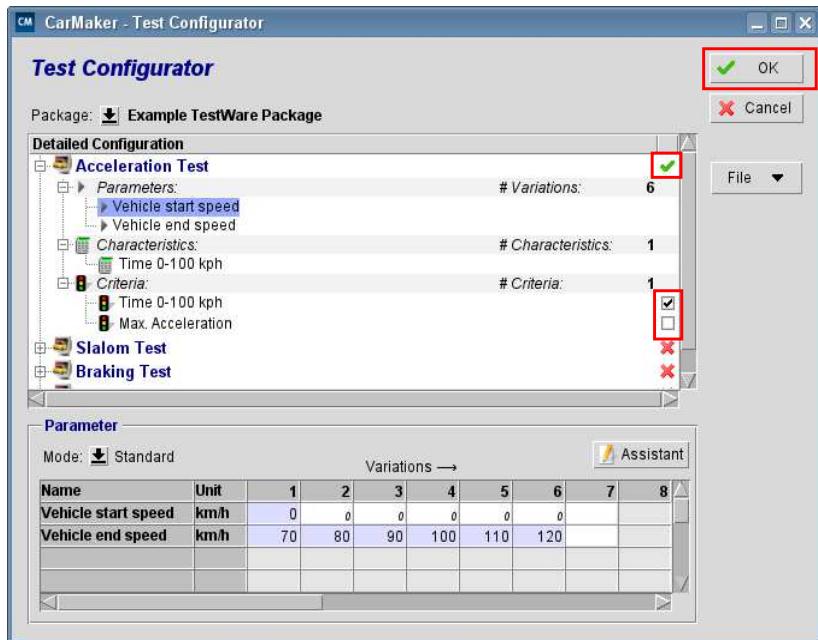


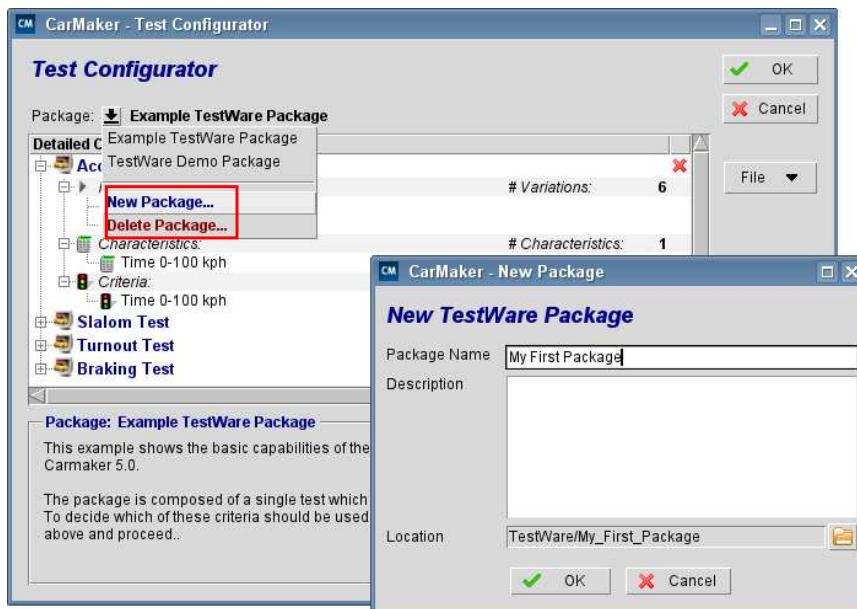
Figure 8.21: Making the test selection and creating a test series

### Creating a TestSeries

Finally, a click to the **OK** button will generate a *TestSeries* out of the specified test configuration. This *TestSeries* is appended as subtree to the *Test Configuration* item currently selected in the Test Manager's main window. To parametrize the test vehicle within the resulting *TestSeries* use the *Vehicle Configuration* items or *Settings* blocks and place them before the actual *Test Configuration* item. The test configuration subtree may also be duplicated to run the tests with different vehicle configurations.

### Managing TestWare Packages

A *TestWare package* basically resides in a subfolder of the *Data/TestRun/TestWare* folder and consists of several files: A *TestWareInfo* file holds some general information about the package and that is mandatory. Each of the package's tests is represented by a *TestWare template* file (\*.tws), accompanied by TestRuns and possible script files.



Adding and removing *TestWare* packages can be achieved via the respective entries in the *package selection* menu on the upper left side of the Test Configuration dialog. Besides managing whole *TestWare* packages it also is possible to modify a single test within a package or add a new one to it. This is done by right-clicking a test item and choosing the appropriate entry from the context menu.

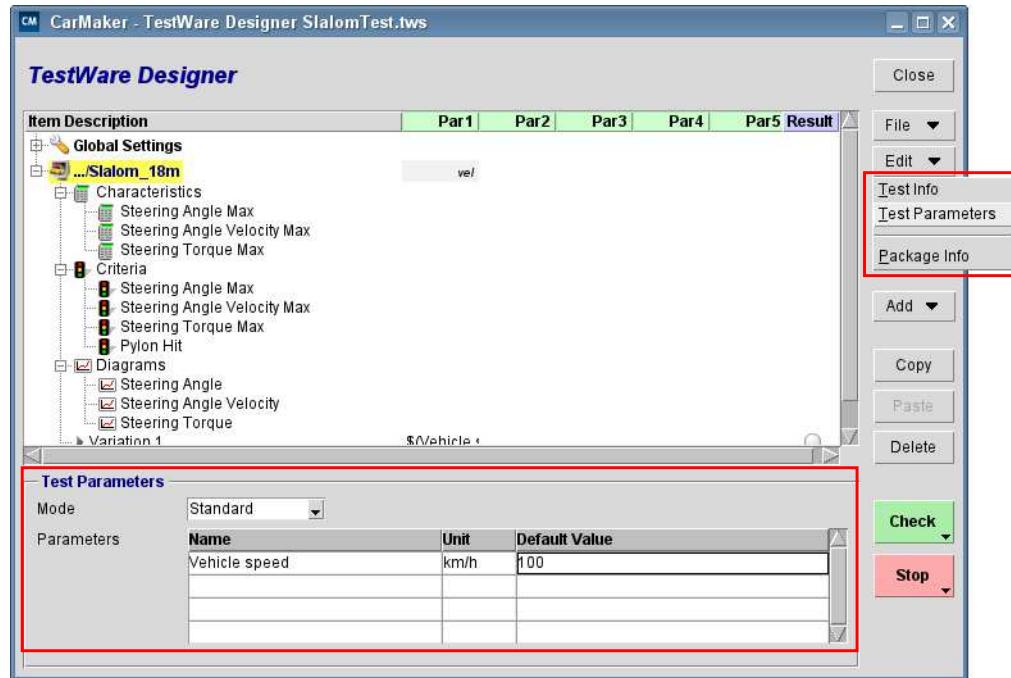


Figure 8.22: TestWare Designer

Tests are created and modified in the *TestWare Designer*, a tool which works on the respective *TestWare* template files. The *TestWare Designer* resembles the *TestManager* window and offers almost the same capabilities, extended only by some additional *TestWare* features. Among those features is the possibility to define test and criteria parameters which later can be modified in the *Test Configurator*. Test parameters are defined by selecting the appropriate menu entry below the *Edit* button. After the definition they can be used in every *Settings* block or *Variation* as place holder for future values. Accordingly criteria parameters are defined on the *Parameter* tab of the respective *Criterion* item and can be used in its evaluation formulas.



A *TestWare* example package consisting of a single test with a few criteria can be found at Examples > CarMakerFunctions > TestWare > DemoPackage.

### 8.1.5 CarMaker Test Report

The *Test Manager* includes a report functionality which automatically generates a report of the performed test series. This report can be called either when a test series is completed or aborted by pressing the Report button in the lower right corner of the *Test Manager* window. Two options are available: Generation of a Test Report for the whole test series (all) or

only for the marked selection (*selected*). The report provides detailed results of every single TestRun clearly arranged. The usage of the Test Manager itself remains completely unchanged.

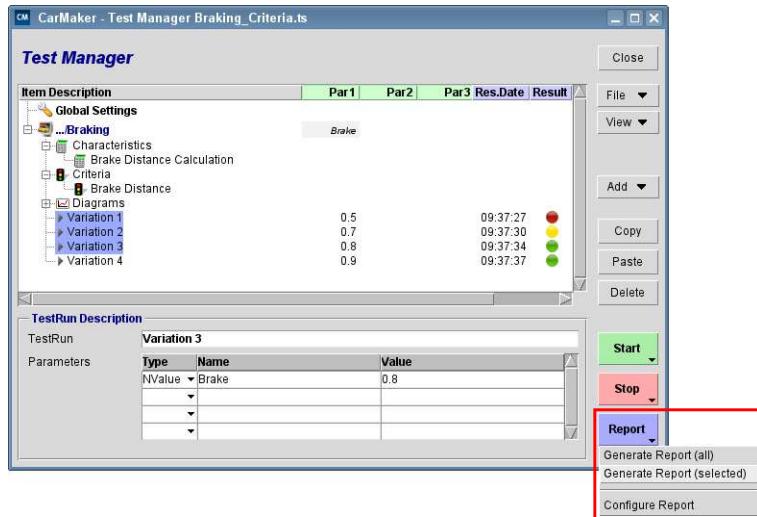


Figure 8.23: Generating a Test Report

## Configuring the CarMaker Test Report

By holding down the *Report* Button on the Test Manager GUI, an additional option called *Configure Report* will appear (see [Figure 8.23](#)). This option enables you to define detailed settings regarding the Report File.

On the first tab *General* of the *Report Settings* dialog you are able to select the output format and path of the generated Report File. Furthermore, you can enter general information like the title, the name of the author as well as remarks.

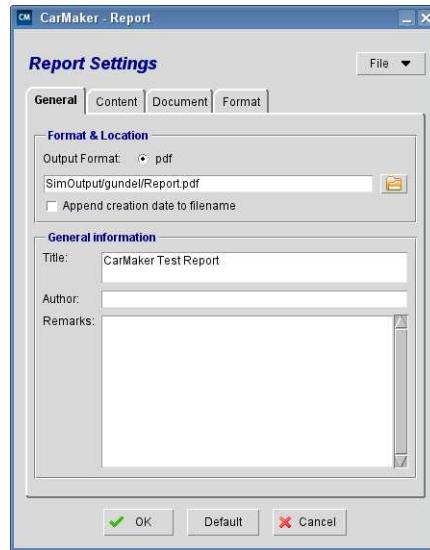


Figure 8.24: Report Settings - General

The second tab *Content* is to customize the content of the report. First of all you are able to select which TestRuns of the test series should be covered by the Report File. Furthermore you can define, which details of the TestRuns will be displayed within the final Report File.

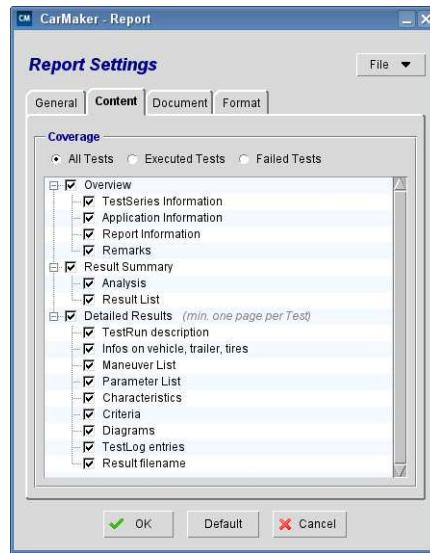


Figure 8.25: Report Settings - Content

After that, the third tab *Document* enables you to set up the orientation, margins, header and footer of the report.

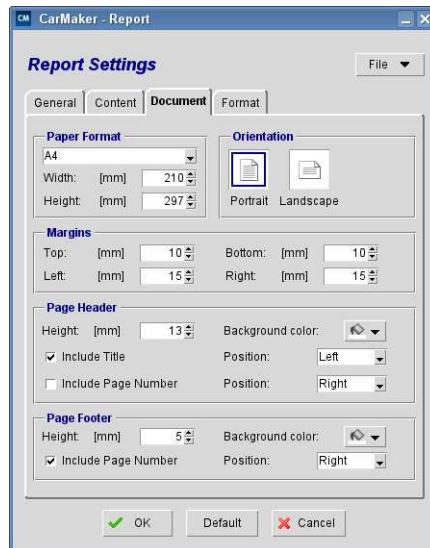


Figure 8.26: Report Settings - Content

On the last tab called *Format*, you can define the font and the colors used in your Report File. Finally, you are able to edit the predefined template using the Report Template Editor. Using the button *Check script*, it is possible to check if the entered script is syntactically correct.

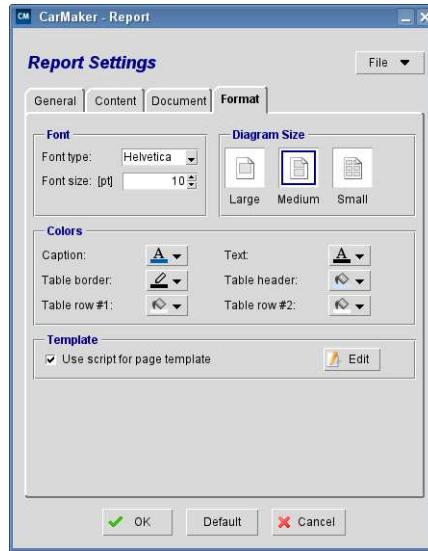


Figure 8.27: Report Settings - Format

## Reading the Report File

The first paragraph within the Report File contains information on the test series itself like its name, starting time and duration. In the second paragraph you will see, beside others, information on the version and path of your CarMaker installation, your project directory as well as a hyperlink to a session log file. The third paragraph contains information on the Report File itself, like the name of the author as well as the date of creation. If you have specified some remarks in the configuration menu before, you will see them underneath the third paragraph.

Within the Report File, you will find the results of your test series, which you have configured in the step before. For more detailed results of a specific TestRun select the respective ID#No. The next chapter gives you an overview of the test results, arranged in a short table.

## 8.2 ScriptControl

If you would like to use test automation intensively, the ScriptControl hits the spot.

ScriptControl is a tool through which you can conduct all actions just as in the CarMaker GUI, using a simple script language (Tcl/Tk).

But you can do even more: read / write files, write html reports etc. and also couple it with external tools. It is simply the best tool to replace the human action during mass tests, e.g. over night!

The Tcl/Tk code can be written to a common text file and needs to be saved to the Data/Script folder of your project directory. Then, load this file in the CarMaker ScriptControl dialog window (Simulation > ScriptControl) using the button *Open*:



Figure 8.28: CarMaker ScriptControl dialog

Pressing the *Edit* button, the currently loaded script file opens in a text editor and you can make changes. Saving the file automatically updates the loaded script. To run a simulation from ScriptControl press the green *Start* button.

To see how a ScriptControl code should look like and what possibilities it offers, open one of the tcl-files in your project directory under Data > Script > Examples. A detailed explanation on how to write an own script is given in the Programmer's Guide, section "ScriptControl".



As an example take the TestRun Examples > CarMakerFunctions > ScriptControl > Straight\_TrailerSwingingDVA.

Please note, that ScriptControl can also be used to remote control the TestManager and Test Report generation. Please find further information in the Programmer's Guide, section "ScriptControl > ScriptControl Command Reference > TestManager commands / Report commands".

## 8.3 Variable Types

Apart from creating test series which automatically executes several TestRuns one after the other, test automation can also be used to run the different versions of the same TestRun by the help of variables. For parameter variations and settings the following variable types are available for test automation in TestManager or ScriptControl:

Table 8.2: Available variable types in the Test Manager

Type	Name	Description
NV	Named Value	Vary parameter field entries in the GUI, see <a href="#">section 8.3.1 'Named Values' on page 488</a>
KV	Key Value	Vary infofile keys, see <a href="#">section 8.3.2 'Key Values' on page 491</a>
TS	TestSpace variable	Create auxiliary variables which are only available in the Test Manager, see <a href="#">section 8.3.3 'TestSpace Variables' on page 494</a>
CM <sup>a</sup>	CarMaker variable ScriptFile	Special variable to call procedures with kinds: <ul style="list-style-type: none"> <li>• <i>ScriptFile</i>: set path to an external tcl-script which holds user defined procedures, see <a href="#">section 8.3.4 'ScriptFile' on page 495</a></li> <li>• <i>StartProc</i>: will be called before every simulation step in the Test Manager, see <a href="#">section 'Start and End Procedure' on page 495</a></li> <li>• <i>EndProc</i>: will be called after every simulation step in the Test Manager, see <a href="#">section 'Start and End Procedure' on page 495</a></li> <li>• <i>SkipProc</i>: will be called when a simulation step should be skipped, see <a href="#">section 'Skip Proc' on page 496</a></li> <li>• <i>ResultFName</i>: change the storage path and name of the created result files, see <a href="#">section 'ResultFName' on page 497</a></li> </ul>

a. TestManager only

### 8.3.1 Named Values

So called *Named Values* (type: NValue) can be put to any editable parameter field of CarMaker. To introduce a *Named Value*, enter a dollar sign (\$) to a parameter field of your choice. The sign needs to be followed by the name you want the variable to have. Optionally you can give this variable a default value.

**Example**      \$Variable1=10



The default value will be used when you start the TestRun in CarMaker (without the Test Manager or ScriptControl). If you do not specify a default value, the TestRun will only be executable by the Test Manager/ScriptControl.

In the TestManager you can define a Named Value parameter variations by:

```
Type  = NValue
Name  = <name of the variable>
Value = <value to be assigned>
```

In ScriptControl the corresponding command is called:

```
NamedValue set <name> <value>
```



Please pay attention to the spelling of your Named Value: The name of the variable needs to be spelled exactly as in the entry field of the CarMaker GUI (case sensitive)!

With *Named Values*, Matlab workspace variables defined under CarMaker for Simulink, can be referenced, too.

Further, *Named Values* can be used in combination with ReatimeExpressions, e.g. as dynamic end condition for a Maneuver (see [section 'End Condition'](#)) or in Maneuver script commands (see [section 'Realtime Expressions'](#)). To reference a *Named Value* in a Realtime Expression, use a Dollar sign. Even a default value can be specified to be able to execute the command without TestManager or ScriptControl extensions. The default value needs to be written in round brackets and is optional.

**Example** Syntax for a dynamic Maneuver end condition using the Named Value *Vel* with a default value of 60 kph:  
`Car.v < $(Vel=60/3.6)+10`

**Example** Syntax for a Realtime Expression in the Maneuver commands with default value:  
`DM.Steer.Ang=sin($(Amp=20)*180/pi * Time)`

## Using Named Values

The following example shows how a variable called *Amp* for the spring amplification factor of the front axle is used with a default value of 2.0:

**Example** `$Amp=2.0`

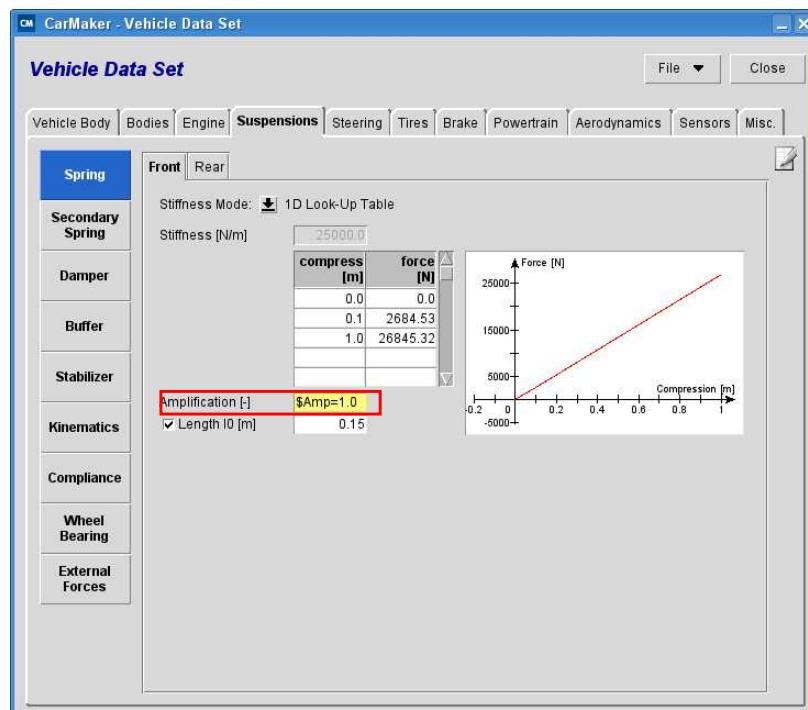


Figure 8.29: Usage of Named Value in the vehicle data set

In the Test Manager dialog you can now refer to the new variable and assign it numerous values:

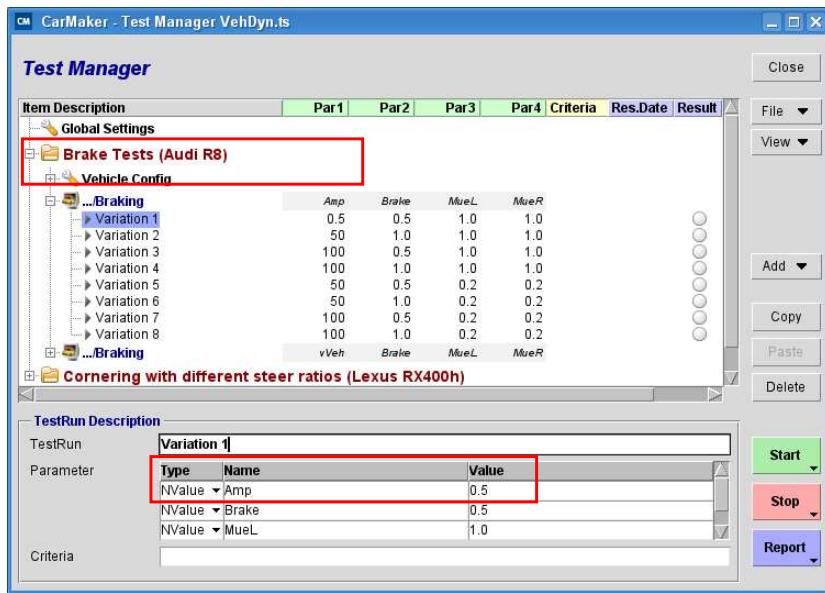


Figure 8.30: Three variations of the Named Value *Amp*

When using Matlab Simulink, a workspace variable is created automatically for each *Named Value*. If a workspace variable with the same name already exists, it will be overruled by the TestManager during the test series execution.

This gives you e.g. the possibility to insert variables to elements used in Simulink to change the behavior of your controller model interactively by the Test Manager.

Apart from a constant value, a vector or a matrix can also be stated in the entry field *Value*.



To use the workspace variable please make sure that you simulate with CarMaker for Simulink instead of CarMaker. Please find more information on the CarMaker interface to Simulink in the Quick Start Guide, chapter 9 "Simulating with CarMaker and Simulink".



An example of the usage of the workspace variable can be found in the CarMaker project folder under Examples/CarMakerFunctions/Test Manager/BodyCtrl.ts. This test series is designed to be used in combination with the Simulink model BodyCtrl.mdl. This model includes a simple roll angle stabilization which applies external suspension forces in dependence of the current lateral acceleration. Within the model extension there is a gain factor which can be varied by the Test Manager. The factor is called *BodyCtrl\_k* and it serves as amplification for the roll angle stabilization. This factor is varied by the help of the Test Manager.

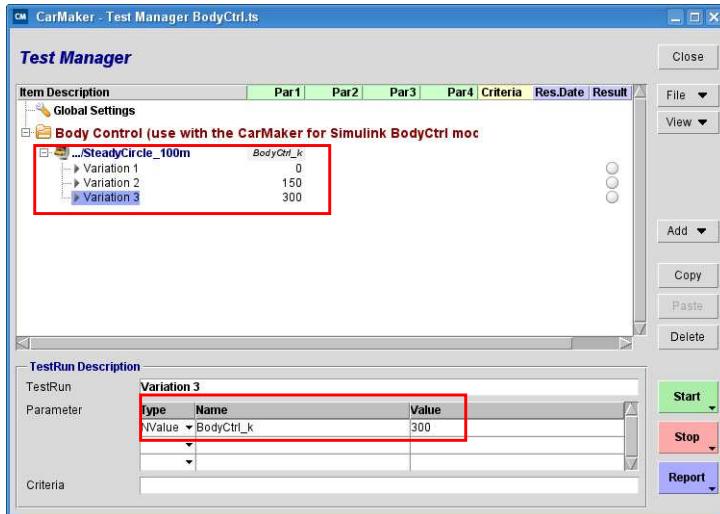


Figure 8.31: Definition of a NamedValue in the Test Manager GUI...

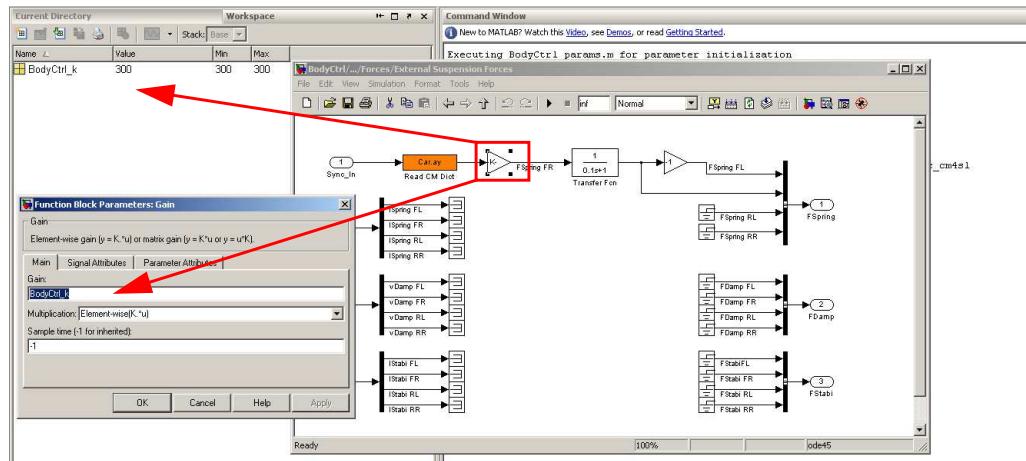


Figure 8.32: ... and its usage in CarMaker for Simulink

### 8.3.2 Key Values

**KeyValues** (type: KValue) are used to modify Infofile keywords which mainly apply for settings that do not have an editable parameter field in the CarMaker GUI. An example would be if you want to perform the same TestRun using different tires or if you want to exchange the vehicle data set.

In the CarMaker GUI the tires are selected directly through a button and a file menu. There is no possibility to enter a *NamedValue* to vary a tire data set.

In that case the corresponding *Key Value* of the TestRun Infofile would be changed. The Infofile itself does not need to be changed, you can do all settings in the variation description. The syntax would look as follows:

```
Type = KValue
Name = <InfofileKind:InfofileKeyValue>
Value = <value to be assigned>
```

In ScriptControl the corresponding command is called:

```
KeyValue set <name> <value>
```

The following Infofile kinds are available:

```
SimParameters
ECUParameters
TestRun
Vehicle
Aero
Brake
Steering
SuspExtFrcs
TireFL, TireFR, TireRL, TireRR, TireRL2, TireRR2, TireFL2, TireFR2
PowerTrain
PTEngine
PTClutch
PTGearBox
PTDriveLine
UserDriver
VehicleControl
Trailer
TrAero
TrBrake
TrSuspExtFrcs
TrTireFL, TrFireFR, TrTireRL, TrFireRR, TrTireML, TrFireMR
```



If you are not sure how the key words are spelled correctly or which key is in which Infofile, you can look directly into the files. All Infofiles can be found in the *Data* folder of your project directory. Further information on Infofiles can be found in the Reference Manual, section 'CarMaker Parameter Files'.

## Using Key Values

If you want to change the front left tire used in your TestRun you would enter the following in the TestManager dialog:

**Example**

```
Type = KValue
Name = TestRun:Tire.0
Value = RT_295_30R18
```

The tires used in a simulation are part of the TestRun definition. For that reason the information on the tire data used for the front left tire is stored to the TestRun Infofile. This leads to the syntax

```
TestRun:Tire.0
```

The Infofile kind is *TestRun* and the key to be changed is called *Tire.0*.

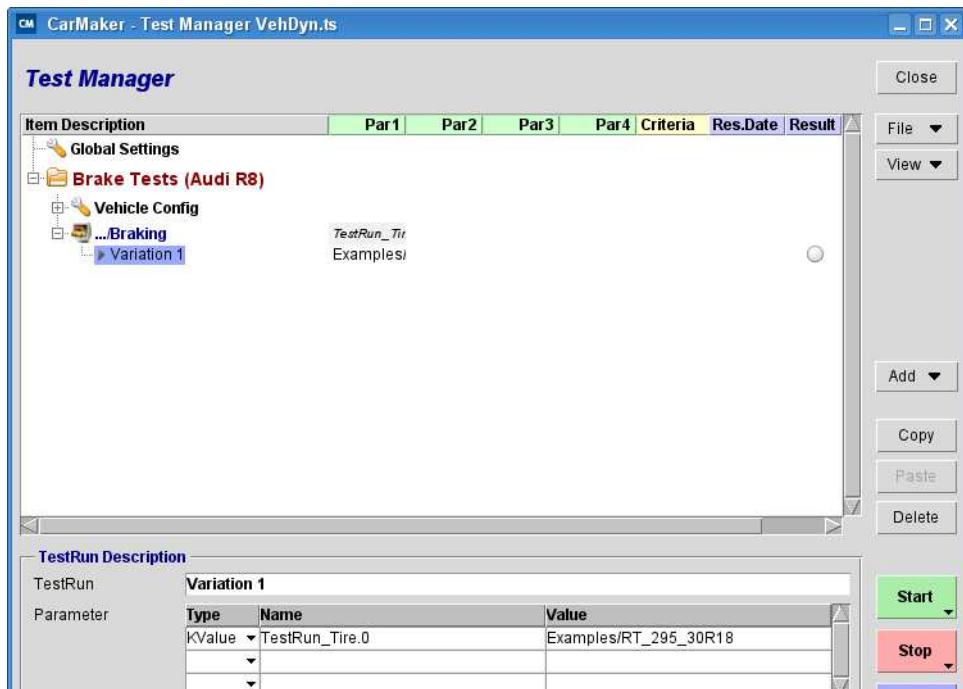


Figure 8.33: Exchanging the front left tire of the vehicle using KValues

If you like to change a tire parameter, e.g. the radial stiffness, you would need to access the Infofile of the tire instead of the TestRun Infofile:

**Example**

```
Type = KValue
Name = Tire:Radial.Stiffness
Value = 40000
```

The Infofile kind in this example is *Tire*, the key to be changed is called *Radial.Stiffness* and its new value is 40000.

As an example take the TestRun Examples > CarMakerFunctions > Test Manager > VehDyn.ts.



### 8.3.3 TestSpace Variables

TestSpace variables are auxiliary variables that are only known within the test series. The functionality is comparable with a blackboard. TestSpace variables can store and calculate values. An overview of all existing TestSpace variables can be retrieved from the View > TestSpace menu in the Test Manager:

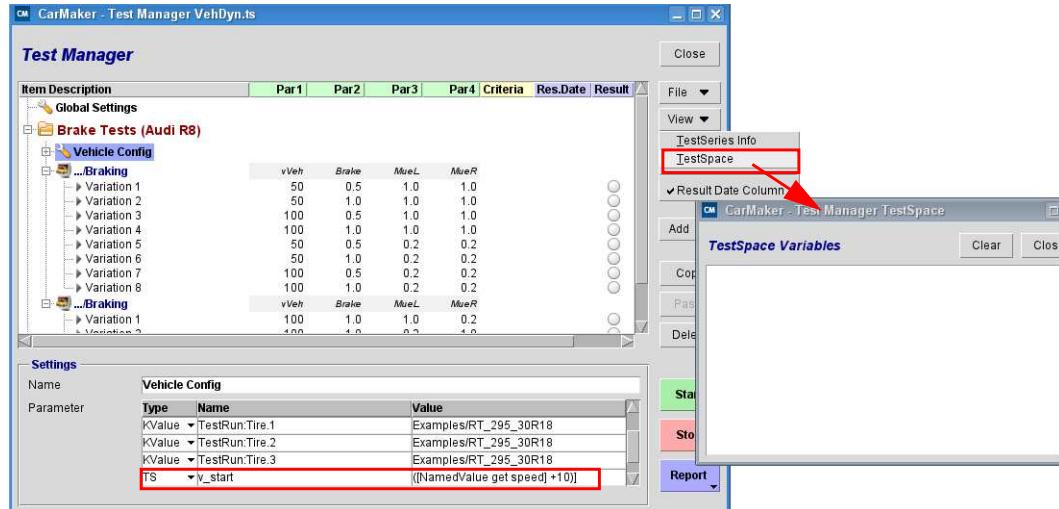


Figure 8.34: Overview of TestSpace variables in the Test Manager

A TestSpace variable can be created in by using

- a *Settings* block in the TestManager GUI:  
with Type = TS, Name = Name of the variable, Value = assign/calculate a value
- a *ScriptControl* block in the TestManager GUI or directly in a Tcl/Tk script using the following syntax:

```
set TS::NameOfVariable Value
```

The TestSpace variables can be accessed using the following syntax:

```
$TS::NameofVariable
```

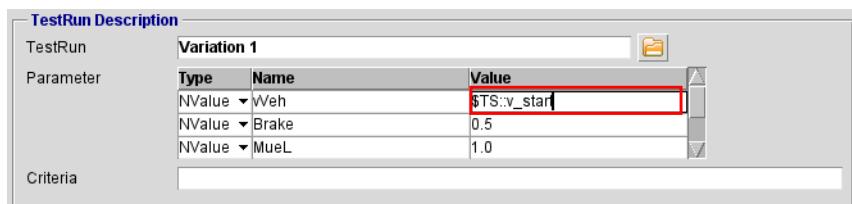


Figure 8.35: Assigning a TestSpace variable to a NamedValue in TestManager



Basically, TestSpace variables are Tcl variables assigned to a special namespace ("TS"). So, there are no special ScriptControl commands available to operate TestSpace variables as all native tcl/tk commands can be used, e.g. `unset TS::NameOfVariable`.

## 8.3.4 ScriptFile

Variables of kind *Script File* only exist in TestManager. They give access to ScriptControl extensions inside the TestManager. Thus, a script file written in the Tcl/Tk programming language can be loaded into the test space so that user defined functions and procedures can be called during the execution of the test series. The script interface already includes three special procedures, which are presented in the following:

### Start and End Procedure

Start and end procedures can be called at all instances of a test series. The following levels are available:

Table 8.3: Hierarchy levels for start and end procedures

Level	Meaning
TestSeries	Execution of whole test series
Group	Execution of a group folder, if defined
TestRunGroup	Execution of a new TestRun level
TestRun	Execution of a variation

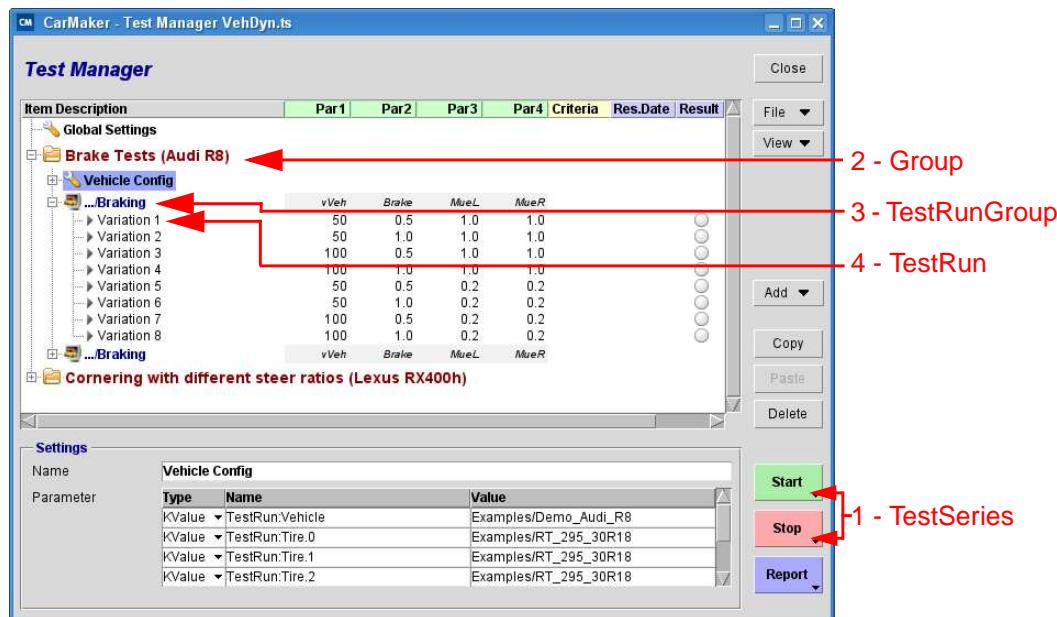


Figure 8.36: Hierarchy of a test series

For an example of a start and end procedure please have a look at the TM\_Procs.tcl file in the Data/TestRun/Examples/CarMakerFunctions/Test Manager/ScriptControl folder of your project directory. The script writes a text output to the logging console of the ScriptControl dialog (CarMaker main GUI > Simulation > ScriptControl). The action commands (in this example the *Log* command) are placed in the curled brackets. The script should serve as a base where you can implement your own code to the curled brackets:

```
proc MyStartProc { key args } {
    set name [lindex $args 0]
    switch $key {
        TestSeries      {Log "MyStartProc TestSeries: $name"}
        Group          {Log "MyStartProc Group: $name"}
        TestRunGroup   {Log "MyStartProc TestRunGroup: $name"}
        TestRun        {Log "MyStartProc TestRun: $name"}
```

```

        default      {Log "MyStartProc unknown key '$key'"; TestMgr::StopTestSeries}
    }
}

```

In the TM\_Procs.tcl there are two procedures defined: the first called *MyStartProc* and the second called *MyEndProc*. To call these procedures in the Test Manager the [Settings](#) should be defined as follows:

Table 8.4: Calling the start and end procedure of the TM\_Procs.tcl script

Type	Name	Value
CM	ScriptFile	ScriptControl/TM_Procs.tcl
CM	StartProc	MyStartProc
CM	EndProc	MyEndProc



As an example take the TestRun Examples > CarMakerFunctions > Test Manager > Slalom.ts.

### Skip Proc

The SkipProc is quite useful to save some test time. You can abort the execution of a test series once a certain condition is met. The function call for the SkipProc is

```
TestMgr::SkipToEnd Kind
```

where kind can be *TestSeries*, *Group*, *TestRunGroup* or *TestRun*, depending on what you want to be left out. The SkipProc can be included to an EndProc, e.g. like the following example shows:

```

proc MyEndProc { key args } {
    set name [lindex $args 0]
    set i 0;
    switch $key {
        TestSeries      {}
        Group          {}
        TestRunGroup   {}
        TestRun        {
            incr i
            if {$i > 0.5} {
                ::TestMgr::SkipToEnd Group
            }
        }
        default        {#Log "MyEndProc unknown key '$key'"; StopTestSeries}
    }
}

```

## ResultFName

This is a special command which influences the storage path and name of the result files created in this test series. In the field *Value* simply enter the new storage path as absolute path or relative to the CarMaker project folder. All result file name macros from the OutputQuantities dialog ([section 'Storage Path' on page 385](#)) are available.

**Example** All result files will be stored in the new folder SimOutput/TestSeries of your CarMaker project directory. The files name reflects the name of the variation:

Type: CM

Name: ResultFName

Value: SimOutput/TestSeries/%v

This command overrules the project settings defined under Application > OutputQuantities.

Pressing the *Edit* button, the currently loaded script file opens in a text editor and you can make changes. Saving the file automatically updates the loaded script. To run a simulation from ScriptControl press the green *Start* button.

To see how a ScriptControl code should look like and what possibilities it offers, open one of the tcl-files in your project directory under Data > Script > Examples. A detailed explanation on how to write an own script is given in the Programmer's Guide, chapter 15 "ScriptControl".

As an example take the TestRun Examples > CarMakerFunctions > ScriptControl > Straight\_TrailerSwingingDVA.



## 8.4 CarMaker Remote Control

CarMaker can also receive commands from external applications which enable automated test driving. The commands need to be based on ScriptControl or the programming language Tcl/Tk. There are several ways to communicate with CarMaker using:

- TCP sockets
- DDE interface on Windows
- Tcl *send* command on Linux
- CarMaker for Simulink command *cmguicmd*

More detailed information on that topic can be found in the appendix of the Programmer's Guide, section "Remote GUI Control".

## Chapter 9

# Miscellaneous

## 9.1 Instruments Panel

The Instruments panel in CarMaker offers a view inside the cockpit to better observe the driver's actions. To start the Instruments dialog, go to *File > Instruments* in the main GUI.

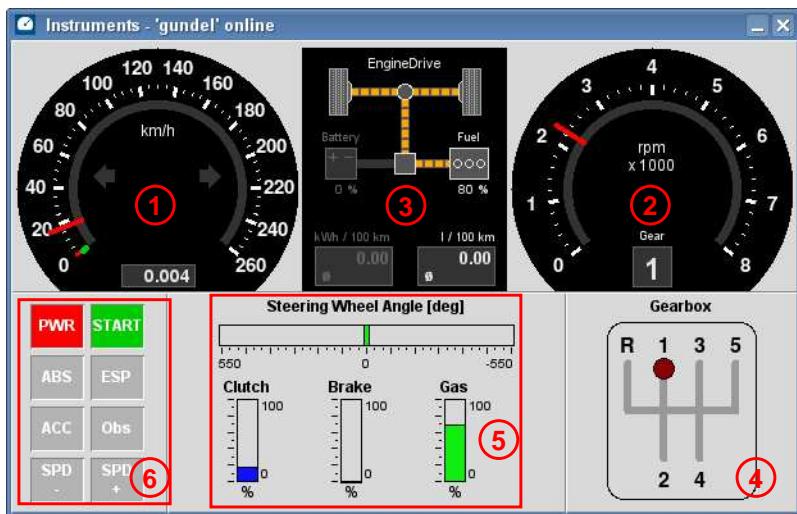


Figure 9.1: The standard Instruments panel

The window shows a common dashboard output such as:

- tachometer (1)
- rev counter (2)
- fuel consumption (3)
- energy flow (in case of hybrid powertrain) (3)
- current gear number (2) + (4)

Further, the driver's steering wheel movements and pedal actions are displayed in section (5) of [Figure 9.1](#).



Please note that this output is based on the User Accessible Quantities of group *DM.\**. In case of a DVA manipulation of the driver's input, e.g. through the Vehicle Control interface, it is still the DrivMan signals which are displayed in the Instruments panel which may not serve as final input to the vehicle model! For further information see [section 6.3 'DVA: Online Manipulation of the Simulation'](#) in this document or section "The main cycle explained" in the Programmer's Guide.

The control lights in section (6) of [Figure 9.1](#) show the current vehicle status regarding ignition (*PWR*) and engine switch (*START*), as well as the availability of vehicle control systems such as ABS, ESP and ACC. A green light indicates the readiness of a system, a red light means the system is deactivated and a grey box implies the vehicle model does support this module.

In case a relevant obstacle (traffic object) is detected by the ACC controller, the "Obs" field turns red.

Some lights can also be used as interactive buttons to turn on / off a system:

- *PWR* for the ignition
- *START* for the engine
- *ESP* for the ESP controller (only available with the CarMaker for Simulink example *HydBrakeCU\_ESP.mdl* or a user defined model extension)
- *ACC* for the active cruise control (only available with certain example vehicles such as the *Demo\_BMW\_5* or a user defined model extension)
- *SPD -/+* to tune the target speed for the example ACC controller implemented in Car-Maker

The Instruments panel is written in the programming language Tcl/Tk. The open source code is available in the CarMaker installation directory under `../IPG/hil/<architecture-version>/bin/Instruments` and can be modified by the user.

Several customized Instruments panels can be used at the same time. For this, save the code to the *bin* folder of the CarMaker project directory. Each filename needs the prefix "Instruments". In the *File* menu of the CarMaker main GUI, all available Instruments panels which follow this naming convention are displayed. The selection menu indicates, if it is a customized Instruments code (label *Project* or *Shared* if source is a data pool), or if it is the standard Instruments panel from the CarMaker installation directory (label *Installation*).

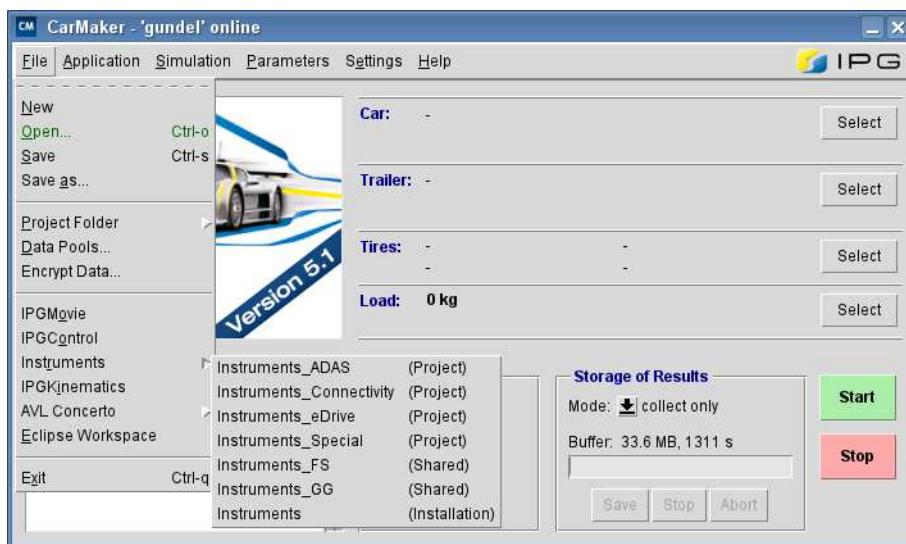


Figure 9.2: Multiple Instruments panels in the selection



Please find several example codes for Instruments panels in the customer area on our website [www.ipg-automotive.com](http://www.ipg-automotive.com).

## 9.2 Data Set Management

### 9.2.1 File Formats

The CarMaker application needs parameter files and datasets. For this purpose CarMaker uses keyword oriented parameter files called Infofiles. They are plain ascii text files but a specific syntax is used. There are Infofiles hidden behind all the CarMaker graphical interfaces that allow you to edit the model parameters: you can modify a parameter either with the graphical user interface or by editing directly the corresponding Infofile.

### 9.2.2 Infofile Syntax

Infofiles use keywords which can be followed by

- a "=" character and several values up to the end of line,
- a ":" character and a multi line entry in the following lines. Each line must start with a <tab> character.

**Example**

```
Body.mass = 1301.  
SuspF.Spring:  
    -0.01 -240.0  
    0.0 0.0  
    1.0 24000.0
```

### 9.2.3 FileIdent

To recognize any changes and incompatibilities of parameter formats and norms (e.g. using a new CarMaker version with out of date datasets) an identification key for each file is used. The identification key shows the model class, the submodel class and the current version.

This identification key is called *FileIdent* and has to be specified in the first line of a parameter file.

**Example**

```
FileIdent = CarMaker-Car 4
```

## 9.3 Data Encryption

With its well-defined file structure and its Infofiles, CarMaker allows its users to easily exchange simulation files and data. For example, to simulate with another vehicle, you just need to copy the corresponding vehicle Infofile in the folder Data > Vehicle. As Infofiles are basically ASCII files, sharing an Infofile implies that you give full access to the set of model parameters which is contained in the Infofile. In some cases, you may want to avoid sharing such data due to confidentiality reasons. In that case you can use the Encrypt Data functionality of CarMaker. It encrypts the content of the files you share to ensure data confidentiality in your exchanges.

CarMaker offers you the possibility to generate and work with encrypted files. Thus, the confidentiality of your data does not prevent you from exchanging them anymore. To generate encrypted files, you can access the *Secured Data File Generator* with the menu File > Encrypt Data...

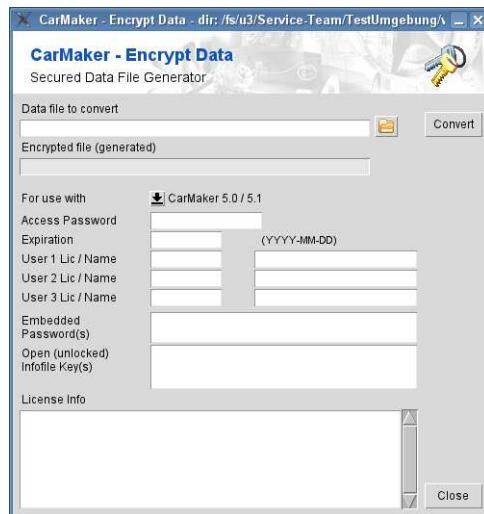


Figure 9.3: Secured Data File Generator

### 9.3.1 Description

This tool allows you to encrypt any CarMaker Infofile. You just have to select the desired file, select the CarMaker version the secured file will be used with and push the button *Convert*. The field *Encrypted file (generated)* indicates you the name of the generated encrypted file: it has the same name as the original file with "@" appended to the end. In this way, it becomes easy for you to recognize at once the encrypted file and its original file.

The *Secured Data File Generator* produces fully encrypted files, e.g. it is not possible to encrypt only a part of it.



You should pay attention to parameter file dependencies: all the parameters for a given system may be contained in more than one file. This is for example the case for the DemoCar. Encrypting the file DemoCar will not secure all the parameters of all its subsystems. Indeed, the parameters of the brake model are not contained in the file DemoCar but in the file HydESP\_DemoParam which is located in the folder Data > Misc. Only a reference to this parameter file can be found in the file DemoCar. That's why you should pay attention when you want to secure all the parameters for a given system. You must first encrypt the files that are referenced in the system Infofile, modify these references (add "@" to the end of the referenced file so that the encrypted file is referenced) and finally secure the Infofile containing the references to the encrypted files.

Using an encrypted file may lead to restrictions in the use of CarMaker. For example, using an encrypted vehicle file will prevent you from using the *Model Check* utility. Editing the data set with the *Vehicle Data Set Editor* (or with a conventional text editor) is not possible, either.



There are a few parameters in the vehicle infofile, which can not be encrypted because they must be readable at the start of the simulation, e.g. for the animation in IPGMovie or CarMaker for Simulink. These parameters are described in the vehicle infofile which should be converted within the section *crypto.show* and hereafter:

```
Picture.PicFName
Movie.Skin.FName
Vehicle.OuterSkin
nAxe
Steering.Kind
Steering.FName
```

```

Brake.Kind
Brake.FName
PowerTrain.Kind
PowerTrain.FName
CM4SL.ActivateFcn
CM4SL.StartFcn
CM4SL.StopFcn

```

Apart from this, the description is always visible.

### 9.3.2 Kind of protection

In addition to the standard encryption, there are several ways to restrict the use of your encrypted files. These protections can be used separately but they can also be combined together to increase the use restrictions.

**For use with** The encryption mechanism and the Infofile layout have changed between different CarMaker versions. It is important to select the correct CarMaker version, which used at the target system. Otherwise, the person receiving the encrypted file will not be able to use it.

**Password(s)** First of all, you can protect your encrypted files with a password, simply by typing it in the field *Access Password*. By clicking on *Convert*, an encrypted file protected with a password will be generated. When loading the generated file, you will be asked to enter the corresponding password:



Figure 9.4: Password asked for a secured file

In the field *Embedded Password(s)*, you may list all the passwords that might have been used to secure the parameter files which are referenced in the Infofile to be encrypted. As a consequence, CarMaker will ask you to enter only one password (and not one for each encrypted subsystem).

**Expiration** To limit the use of your encrypted file in time, you have the possibility to set an expiration date in the following format YYYY-MM-DD, e.g. 2010-12-31. Once the expiration date has run out, an error message will indicate you that your file cannot be used anymore.

**User/License** Another way to protect your files is to restrict their use to a given CarMaker license. You can enter up to three license numbers. To find the license number that is requested here, please select Help > License Info > Details in the CarMaker main GUI and look for *license serial #*. The field *Name* can be used to enter the user name as owner of the CarMaker license. However, the name entered does not affect the usage if the file.

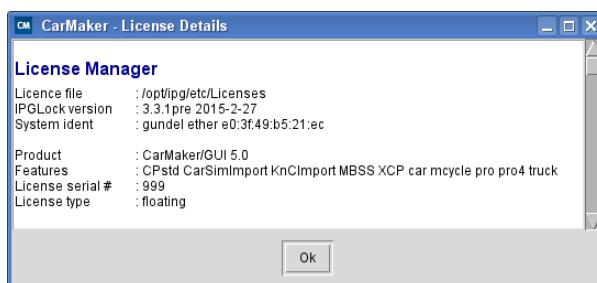


Figure 9.5: Obtaining the license serial number from the license info

The serial number tagged *Serial/No* is the number required in the field *User No. Lic* to limit the use of the generated encrypted files with this CarMaker license. In the field *User No. Name*, you can simply enter the name of the user(s) who will have the permissions to work with your files.

- Open (unlocked) Infofile Key(s)** Some infofile keys can be made available to the receiver, e.g. to get some general information about the parameter set or as simulation input for second party software. If your vehicle model provides e.g. some parameters for a controller model which run in Matlab/Simulink, you can make these parameters accessible for the co-simulation. However, all other infofile keys are still encrypted and unreadable with a text editor.
- License Info** In the *License Info* area, you can enter any useful information about the encrypted file. The information will be displayed in CarMaker when the file will be loaded.

## 9.4 CarMaker GUI Settings

The menu *Settings* allows you to customize the CarMaker application.

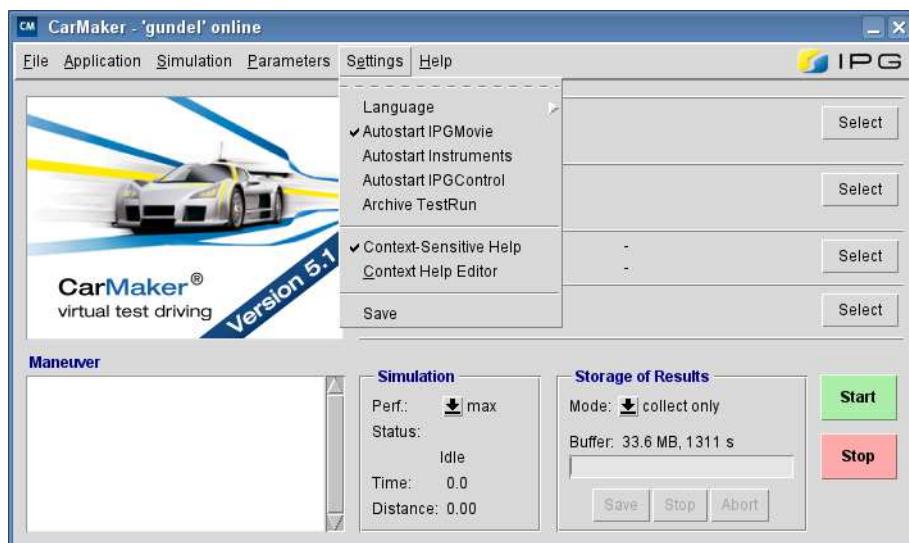


Figure 9.6: Menu Settings

- Result Folder** As indicated in [section 6.5.2 'Configuring the Saving Process' on page 383](#), you can choose the directory where your simulation results will be stored.
- Language** CarMaker is available in two languages: English and German. This item makes it possible for you to switch from one language to another. All labels in the graphical user interfaces will then be displayed in the selected language. For the changes to take effect, the application needs to be restarted. You can do it right away or wait for the next start of the application.
- Autostart** With the items *Autostart IPGMovie*, *Autostart Instruments* and *Autostart IPGControl*, you can select the tools that will be started automatically each time you open the CarMaker Main GUI.
- Archive TestRun** As indicated in [section 6.7 'CarMaker Archive Service' on page 395](#), this is the way to activate the CarMaker archiving functionality.

**Context Sensitive Help** In some graphical user interfaces, a balloon help message can appear when the mouse cursor is placed for a while on a given label. The message gives more information about the functionality of the selected button. This item activates / deactivates this functionality.

**Context Help Editor** In addition to the original set of balloon help messages, you can define our own help messages thanks to the following tool:



Figure 9.7: Context Help Editor

First of all, select the Tk widget (e.g. the element of the GUI) in the list that should have a balloon help message. Then, you need to enter your customized help message in the field *User defined*. In the fields *English* and *German*, you can see if a balloon help message was already affected to the selected widget (these fields are read-only).

**Example** As an example, you can define a balloon help message for the Start button of the main GUI by selecting the widget ".f.btn.start". After typing the following text:  
Start\nbutton  
and clicking on File > Save, your message will be displayed in this way:

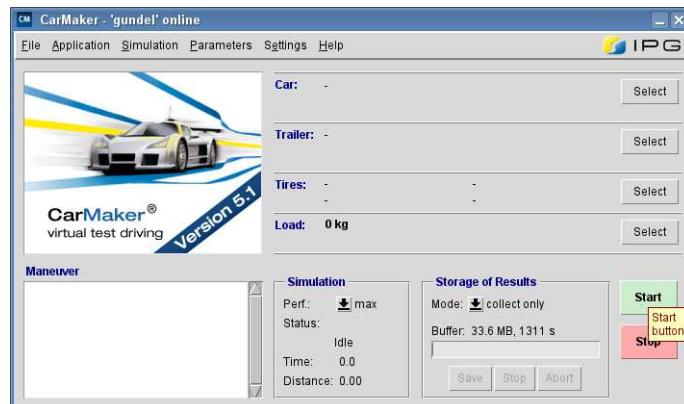


Figure 9.8: Context Help Example

**Save** Saving your Settings will store your preferences in memory so that they will be taken into account at the next start of the CarMaker application.

## 9.5 Documentation / Help

The menu is important as it contains the links to the CarMaker documentation. A lot of answers to the questions you may have about CarMaker can be found in these documents.

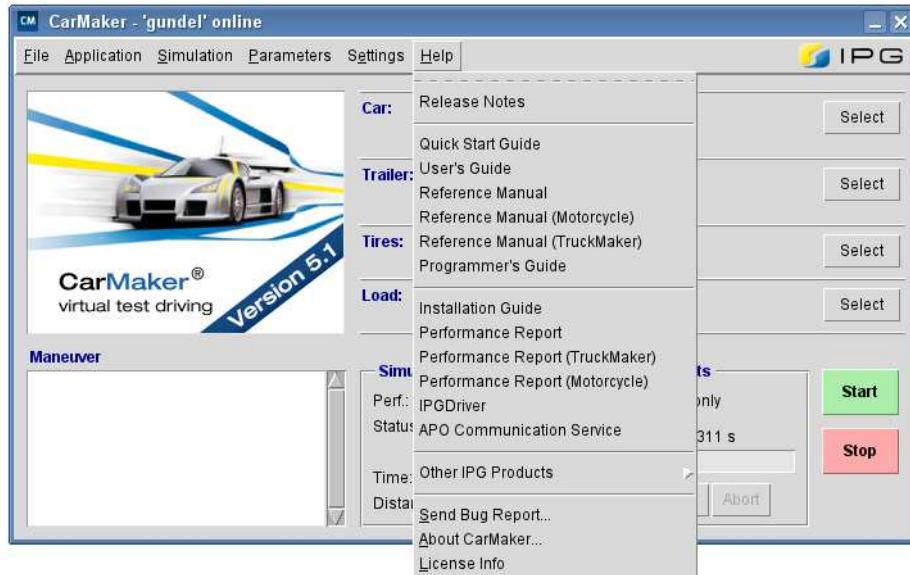


Figure 9.9: Menu Help

- |                           |  |
|---------------------------|--|
| <b>Release Notes</b>      | The Release Notes give an overview of the CarMaker release history, the new features of the application and indications about CarMaker upgrades.   |
| <b>Quick Start Guide</b>  | The Quick Start Guide intends to give a first introduction to CarMaker by showing how to perform simulations and explaining the basic functionalities.   |
| <b>User's Guide</b>       | The User's Guide describes more in details the concepts, the functionalities and the handling of the CarMaker application.   |
| <b>Reference Manual</b>   | The Reference Manual is a document for those who want to get a deeper knowledge of all models implemented in CarMaker. It also includes the list of the User Accessible Quantities, e.g. the variables that can be visualized with IPGControl. The versions "Motorcycle" and "TruckMaker" are add-ons to the CarMaker Reference Manual listing the parameters and model specific for the corresponding software.   |
| <b>Programmer's Guide</b> | The Programmer's Guide contains information about how to customize the CarMaker simulation program to meet new requirements (displaying log messages, reading new parameters in Infofiles, accessing new variables, integrating Simulink Models, managing IO communication for HIL applications, using the FailSafeTester). It also deals with CarMaker in Simulink, the different utilities to be used with Matlab and ScriptControl for test automation. |
| <b>Installation Guide</b> | This document basically describes the installation process of CarMaker as well as the installation and configuration of the real time system with a CarMaker/HIL test stand. Important information about the system requirements to install CarMaker are also contained in this document.  |
| <b>Performance Report</b> | The Performance Report shows a comparison of simulation results between CarMaker 5.1.4 and CarMaker 4.5. The versions "Motorcycle" and "TruckMaker" are add-ons to the CarMaker Performance Report listing the parameters and model specific for the corresponding software.   |

- IPGDriver** A separate manual describes the functionality and parameterization of the IPGDriver model.
- APO Communication Service** The APO Commuication Service is responsible for the data exchange when the simulation is running. The CarMaker GUI, IPGMovie, IPGControl and the Instruments panel communicate via this service with the CarMaker application during a simulation. This document describes the APO service and shows how the user can add other tools to the online communication.
- Other IPG Products** In this menu, you can find user manuals for other IPG products, e.g. IPGKinematics.
- About CarMaker...** This item opens a window showing the GUI version number, legal information about IPG product names as well as contact information.
- License Info** This last item allows you to get license information, e.g. license status, serial number, system identification and location.

### 9.5.1 CarMaker Bug Report

In case you encounter a problem or an error message while using CarMaker, you can send it to the CarMaker Service Team. To be able to fix a bug it is very important, that the CarMaker Service Team can reproduce it. In case you want to report a bug, please always use the CarMaker Bug Report. It collects besides the error message itself other important information so that the problem can be analyzed most efficiently. Of course, you can view and edit all information before it is send out.

The CarMaker Bug Report window can be started either from the bug message itself or from the *CarMaker main GUI > Help > Send Bug Report*.

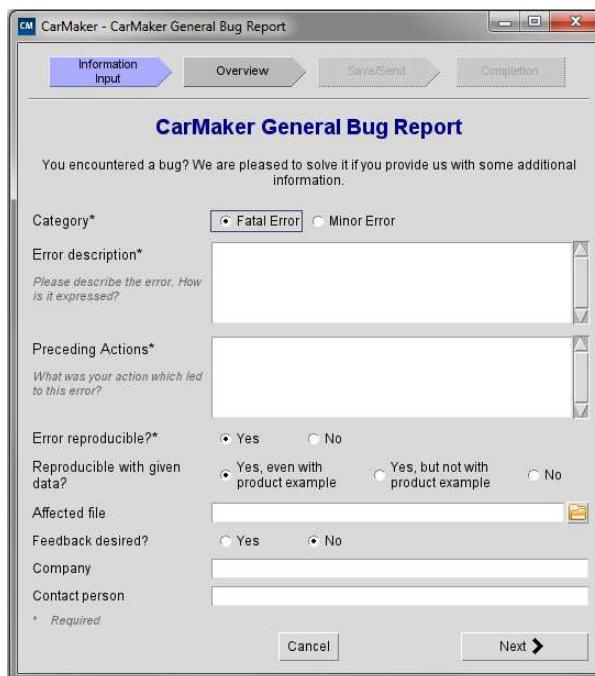


Figure 9.10: CarMaker Bug Report - Information Input

## Information Input

<b>Category</b>	Please inform us if the problem is very critical as it prevents you from work ( <i>Fatal Error</i> ) or if the error only occurs occasionally ( <i>Minor Error</i> ).
<b>Error description</b>	Here, the error message can be pasted or - if missing - you can enter a description of your problem.
<b>Preceding Actions</b>	What actions were taken before the error message appeared? Which TestRun/vehicle data set were used? Which click resulted in the error message?
<b>Error reproducible?</b>	Select Yes if the error can be reproduced. Please describe the exact procedure to generate the error message under <i>Preceding Actions</i> . In case the error is reproducible, you can refer to one of our examples which lead to the error message.
<b>Reproducible with given data?</b>	If Yes is selected at <i>Error Reproducible?</i> , the GUI will be extended by this query. At this point, select the appropriate answer.
<b>Affected File</b>	If Yes is selected at <i>Error Reproducible?</i> , the GUI will be extended by this query. At this point, select the respective file, which leads to the occurred bug.
<b>Feedback desired?</b>	Select Yes, in case you would like to know when the bug is fixed.
<b>Company / Contact Person</b>	Please state your name and you company name in case you desire a feedback.

## Overview

In the next step you get preview of all the information sent to the CarMaker Service Team.

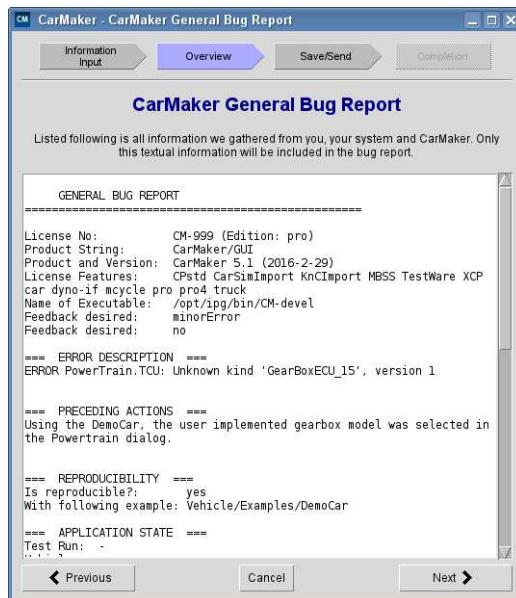


Figure 9.11: CarMaker Bug Report - Overview

## Save / Send

In the next step you can decide if the email should be sent out directly or if you want to save the bug report and send it later. The report will be saved in the CarMaker project directory under *BugReport*.



Figure 9.12: CarMaker Bug Report - Save / Send

## 9.6 Simulation Parameters

In SimParameters all "global" (TestRun, vehicle,... independent) simulation specific parameters are given. The configuration file is available in the CarMaker main GUI under *Application > Edit 'SimParameters'*.

For further explanations, see the Reference Manual, chapter 3 "CarMaker Configuration".

## 9.7 TestRun Parameters

Like additional parameters for a vehicle data set, the user can add additional parameters to a TestRun. For this, choose "Additional Parameters" from the *Parameters* menu in the main GUI.

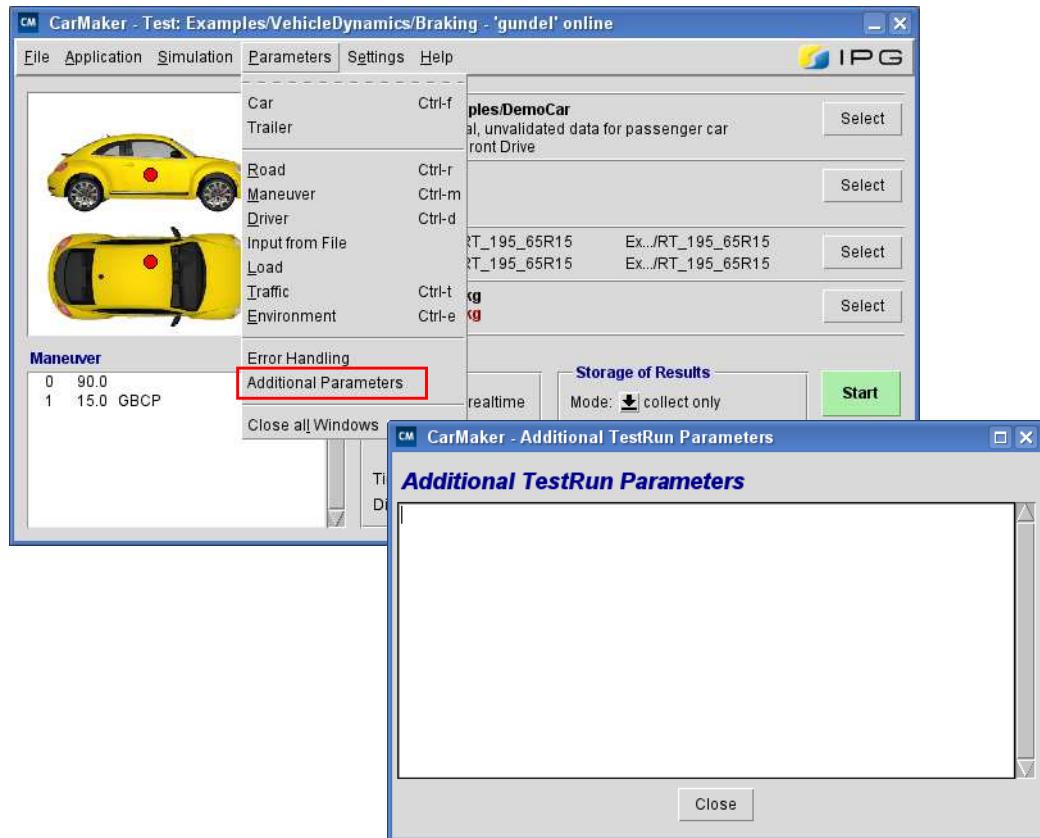


Figure 9.13: Editing Additional Parameters to a TestRun

The *Additional TestRun Parameters* are not required to simulate the CarMaker models. In case of user model extensions, configuration parameters for the user models can be added to the TestRun Infofile as an alternative to the modification of the vehicle data set.

An example is the activation of a *Generic Plugin* model and the definition of its DVA place. A *Generic Plugin* model can be any model which does not have to interact with the vehicle (see Programmer's Guide for further information about *Generic Plugin models*).

## 9.8 Session Log

In the menu *Simulation*, you can open and visualize the CarMaker *Session Log*, e.g. a record of events for each simulation in a log file. Indeed, it is useful to keep a history of important or unusual situations and events during the simulation of a TestRun, that does not disappear when the simulation is finished or the user turns off his computer. This might be of particular interest for test automation, as you can trace any potential errors that might have occurred in a given session.



The Session Log file is limited to 10 MB. If this size is reached, only the official CarMaker events will be written into the file.

For more explanations about the logging module, see the Programmers Guide, [section 'Logging Module'](#).

### 9.8.1 Test Stand Usage

This utility accessible from the menu Simulation > Test Stand Usage gives you an overview of the simulation history for a given system.

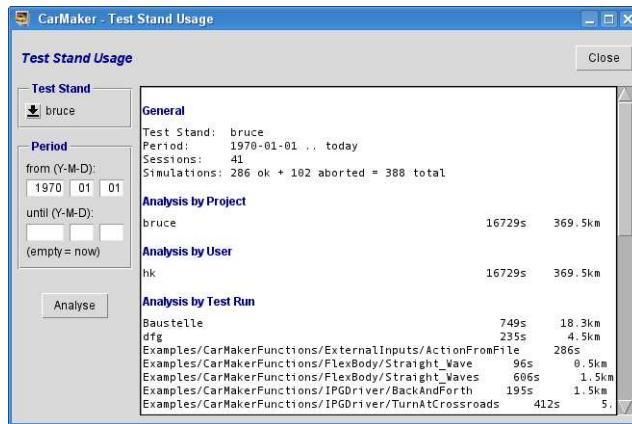


Figure 9.14: Test Stand Usage

**Test Stand** In this list you have to select the system to be analyzed.

**Period** In this area you specify the time frame for the analysis.

**Analyse** Invoking the button *Analyse* will proceed to the system analysis. Its results will be displayed and classified according to the following criteria:

- Project: a list of all projects (defined with Settings > Result Folder...) is displayed with total simulation time and distance for each of them.
- User: all users that have run simulations on the system are listed in this section. Total simulation time and distance for each of them are shown as well.
- TestRun: TestRun files that have been loaded and played in the defined time frame appear in this section with total simulation time and distance.

## Chapter 10

# Connecting to Third Party Tools

## 10.1 Overview

You may want to combine CarMaker's complete simulation environment with a tool that is specialized in a certain task of simulation such as postprocessing and powertrain. The Programmer's Guide introduces different possibilities to integrate a user component model into CarMaker, add functionalities or establish a connection to another tool.

Besides, CarMaker offers pre-established interfaces to some specialized third party tools. How to use CarMaker with postprocessing tools (e.g. AVL Concerto, Matlab) can be found in [section 6.6 'Postprocessing'](#). Additionally, the following third party tools are supported by CarMaker (alphabetic order):

- In the development process of advanced driver assistance systems, CarMaker's interface to **ADTF** (Automotive Data and Time-Triggered Framework) provided by Elektrobit is a crucial feature, e.g. for sensor fusion.
- **AVL Concerto**, as a powerful data analyzation tool can be coupled with CarMaker for postprocessing. Please find further information in this document under [section 6.6 'Postprocessing'](#).
- Powertrains modeled in **AVL Cruise** can be co-simulated with the CarMaker vehicle, driver and environment model (co-simulation).
- **GT-Suite** can be used to build detailed powertrains and test them inside CarMaker's environment (co-simulation).
- Roads including traffic signs, traffic lights etc. can be imported from **Here ADAS RP** into CarMaker. Additionally, preview horizon information can be exchanged between the tools.
- **Matlab from MathWorks** can be combined with CarMaker for several purposes: post-processing (see [section 6.6 'Postprocessing'](#) in this manual), co-simulation (search for "CarMaker for Simulink" in the Quick Start Guide and Programmer's Guide), generation of c-code models (see section Simulink Coder Interface in the Programmer's Guide), remote GUI control (see equally named section in Programmer's Guide).
- **Michelin Tame Tire** contains a sophisticated, physical tire model that can be simulated with the CarMaker environment. Please refer to [Reference Manual chapter "Tire Model Tame Tire"](#) for more information.

- The **Adletec SoundMaker** application is an audio module which generates engine, air and tire noises based on currently running simulation in CarMaker.
- The implementation of the Magic Formula tire model from **TASS Delft-Tyre** (also known as TNO tire) can also be used as tire in CarMaker.

A complete list of all supported third party tools can be found in [Release Notes](#) and on our homepage ([www.ipg-automotive.com](http://www.ipg-automotive.com)).

## 10.2 ADAS: ADTF Interface

### 10.2.1 Overview

For data exchange with ADTF, the ADTF UDP extension provided by ADTF has been used. This allows access to the ADTF message bus using the UDP/IP protocol by ADTF-to-X communication. The ASYNC mode of the default UDP Protocol from ADTF has been implemented.

In the default UDP protocol from ADTF, the UDP packets have a common header followed by the data part. When the ASYNC mode is activated, the data part contains a complete media sample/media stream. The structure of this stream has been implemented according to the "Data Definition Language" (DDL). The DDL contains the description of the data types and data structures used and thus, defines the structure of the transmitted data packages.

The CarMaker ADTF Interface allows the user to perform a mapping between the stream elements and User Accessible Quantities (UAQs). Furthermore a scaling factor can be provided when reading or transmitting the data.

The ADTF configuration dialog can be found in the CarMaker main GUI under *Application > ADTF Configuration*. It allows activation, deactivation and detailed configuration of the data swap with CarMaker.

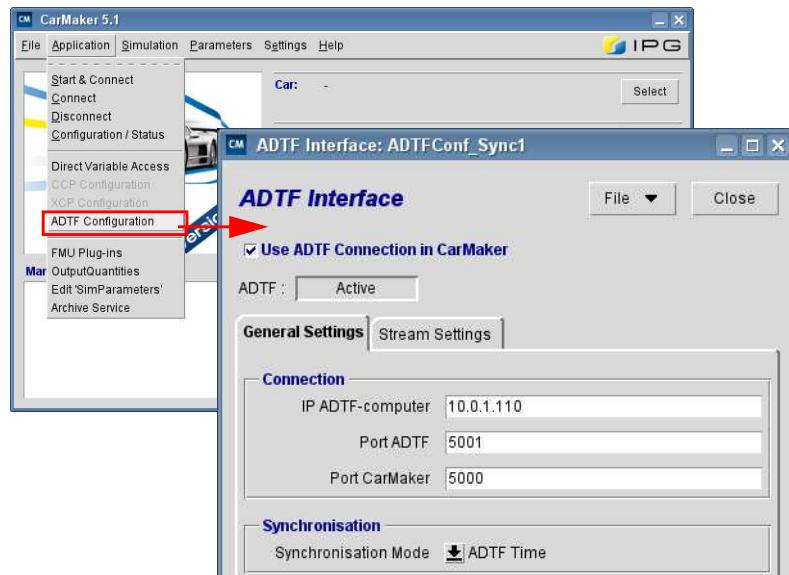


Figure 10.1: Opening the ADTF configuration dialog

## 10.2.2 Configuration GUI

The configuration is done via Infofile parameters that are stored in a separate configuration file. Since the parameters are stored in ASCII format, it is possible to edit all the configurations using a text editor or other external tools or scripts.



It is possible to have different configurations that can be loaded and saved via the *File*-menu. Configuration files must be saved in the project directory under *Data/Config*. The name of each configuration file can be freely assigned. Please note, that changes in the current configuration don't need to be saved to take effect on the next simulation run.

To start the ADTF module, a configuration needs to be loaded and activated. Activation can be performed through the upper left checkbox ([Figure 10.2](#)). Example configuration files can be found in the installation directory under *Examples/ADTF*. The corresponding DDL file *adtf.description* and the associated ADTF configuration are also available there. To use them, copy the configuration files and the DDL file in *Data/Config* of your project directory and adapt at least the connection settings in the configuration file (either with a text editor or with the ADTF GUI).

Menu entries in detail:

- "Open": Select and load an existing configuration file.
- "Save": The current configuration data will be saved.
- "Save As": The configuration data can be saved under a new name.

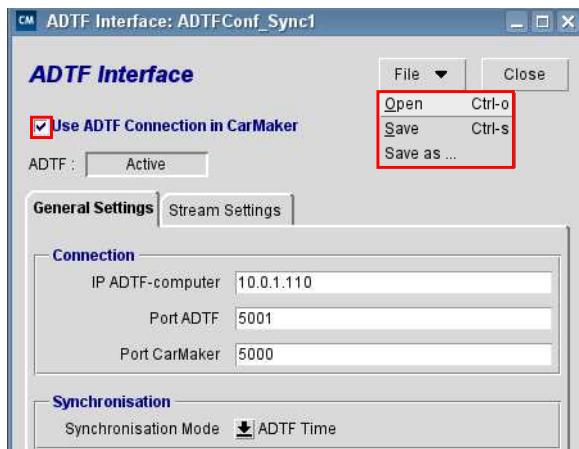


Figure 10.2: File menu options

The current status of the ADTF Interface is displayed in the configuration dialog as follows:

- "Init": Initialization phase
- "Subscribe": Subscription process is being carried out ("subscribe")
- "Active": Active data exchange
- "Unsubscribe": Un-subscription process is being carried out
- "Off": Module inactive or the loaded configuration is not active
- "Unknown": CarMaker isn't running, i.e., the GUI has not been connected to CarMaker or the CarMaker environment has not been extended to include the ADTF Interface

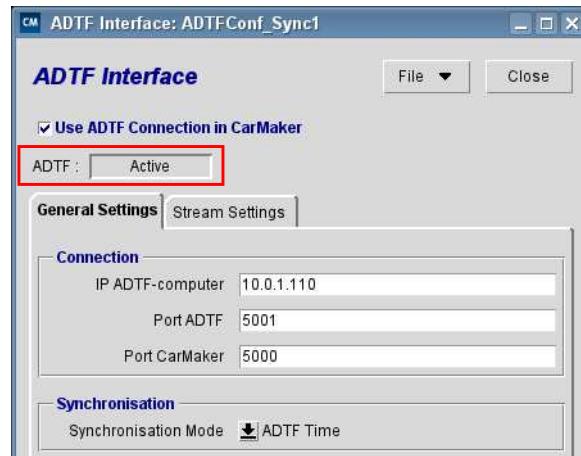


Figure 10.3: Status in ADTF Configuration Dialog

### 10.2.3 General Settings

The *General Settings* tab of the Configuration GUI allows the user to configure the connection to the ADTF system as well as the selected synchronization method.

When the ADTF-Interface is activated, an active connection to ADTF is required to start the simulation. If the connection gets lost, the simulation will proceed.

#### Connection Options

- IP ADTF-computer** Please enter the IP address of the system on which the ADTF is running in this field. The computer system on which ADTF is running must be on the same subnet as the one on which CarMaker will be running on. Broadcast of addresses is allowed.
- Port ADTF/Port CarMaker** The ports for the data exchange have to be mentioned. The data exchange port for the ADTF has to be provided under *Port ADTF* and the port for the received data by CarMaker must be provided in the field *Port CarMaker* respectively. Only one port can be specified within ADTF. The other port is incremented by 1 as shown in the figure below. In case the ADTF and CarMaker simulations are running on the same computer system, the two ports need to be interchanged. In the ADTF environment this is done automatically but in CarMaker, this configuration must be adjusted accordingly.

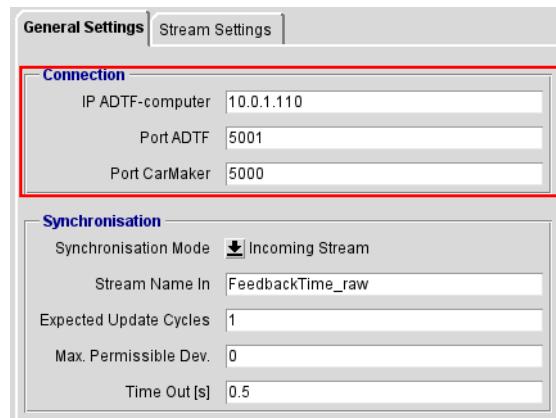


Figure 10.4: Connection configuration of the ADTF Interface

## Synchronization Options

There are different synchronization methods available to synchronize CarMaker and the ADTF simulation. The synchronization can be configured in the "Synchronization"-section of the configuration dialog as shown in [Figure 10.5](#). All synchronization methods are working only in simulation state "Simulate" (in simulation phase "Running").

Choose synchronization mode "ADTF Time" if you want to synchronize the CarMaker simulation with the ADTF time or select mode "In- and outgoing Stream" if your models in ADTF should be calculated between two CarMaker cycles.

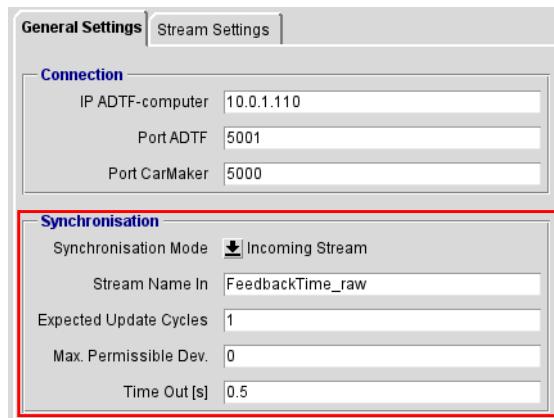


Figure 10.5: Synchronization configuration of the ADTF Interface



Please note that CarMaker should always run faster than the ADTF application when synchronization is switched on. ADTF is the simulation master and will not wait for CarMaker. For that reason the simulation speed is automatically set to "ADTF" and CarMaker will run at maximum speed when synchronization is activated.

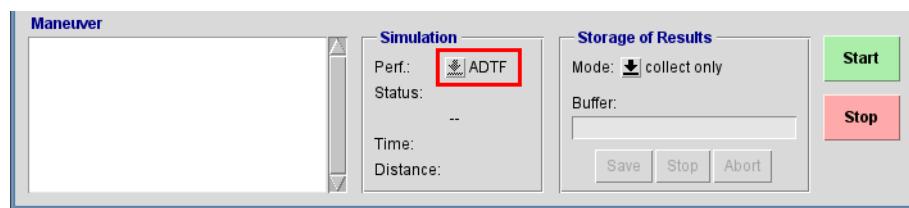


Figure 10.6: When ADTF is active and used with a synchronization mode, the simulation speed depends on the one of ADTF. This is reflected in the main GUI.

### Synchronization Mode

Use the pull-down-menu to select between the following synchronization methods:

- **Off (0):** Synchronization is switched off
- **ADTF Time (1):** The last data from ADTF before state simulate of a new TestRun will be used as time stamp to synchronize the TestRun simulation time in CarMaker with the time in ADTF. ADTF is the time master and the ADTF interface within CarMaker is waiting for ADTF. Further parameters are not necessary for this mode.
- **Incoming Stream (2):** An incoming data stream from ADTF defined by the parameter "Stream Name In" is used to synchronize the CarMaker simulation. In every cycle when an arrival of the defined stream is expected, CarMaker is waiting for it. Thus, the Parameter "Expected Update Cycles" has to be defined correct. To speed up the simulation it is possible to let CarMaker step a bit further. This can be achieved by specifying

the number of cycles CarMaker may continue, even if no stream arrived in the expected cycle, in the field "Max. Deviation". To stop the simulation when CarMaker is waiting for an incoming message, a time out is used ("Time Out").

- *In- and outgoing Stream (3)*: An incoming data stream given by the parameter "Stream Name In" and an outgoing data stream defined by the parameter "Stream Name Out" are used to synchronize. ADTF should be configured, that the data stream given in "Stream Name In" is sent to CarMaker independently on the received stream (given by "Stream Name Out").

CarMaker then waits in each cycle until the number of sent streams is equal to the one of the received streams. Again it is possible to speed up the simulation. One way is to specify a permissible deviation between the numbers of streams sent and received in the field "Max. Deviation". So CarMaker will only wait for the next incoming stream, if this gap is exceeded. Another way is to check the equality only on each x-th sent stream (limit the number of validations). Enter the number for the 'x' in the field "Step".

So with the first variant one gets a continuous deviation between the number of messages, and with the second one the number of messages is equalized after each x-th sent message. It is possible to combine the two methods.

As in mode 2 a time out is given by the parameter "Time Out". The counters are reset when starting a new TestRun.

- *Sync Signal (4)*: A special synchronization variable (of type unsigned integer) is used. This variable is counted each cycle. So the ADTF interface within CarMaker is waiting for the update. The stream sent by CarMaker should be sent each cycle and ADTF should send back a resulting stream immediately. The Parameter "Signal In" and the Parameter "Signal Out" are specifying a data variable within the streams to and from CarMaker. The addressed variables need a size of 4 bytes. A time out given by the parameter "Time Out" is used.

**Stream Name In** Modes: 2, 3 Name of a data stream with direction from ADTF to CarMaker.

**Stream Name Out** Mode: 3 Name of a data stream with direction from CarMaker to ADTF.

**Expected Update Cycles** Mode: 2 Specifies the expected number of CarMaker cycles which are necessary until the given stream (*Stream Name In*) will be received again.

**Max. Deviation** Modes: 2, 3 Maximum Permissible Deviation: tolerates a delay by the given number of CarMaker cycles (in mode 2) or accordingly tolerates a deviation by the given number of sent and received messages (in mode 3). (Improves the simulation speed.)

**Step** Mode: 3 A synchronization will be done only each x-th communication cycle.

**Signal In** Mode: 4 Name of a variable within a received data stream for the synchronization counter.

**Signal Out** Mode: 4 Name of a variable within a sent data stream for the synchronization counter.

**Time Out** Modes: 2, 3, 4 Time in seconds used as time out in a CarMaker waiting phase

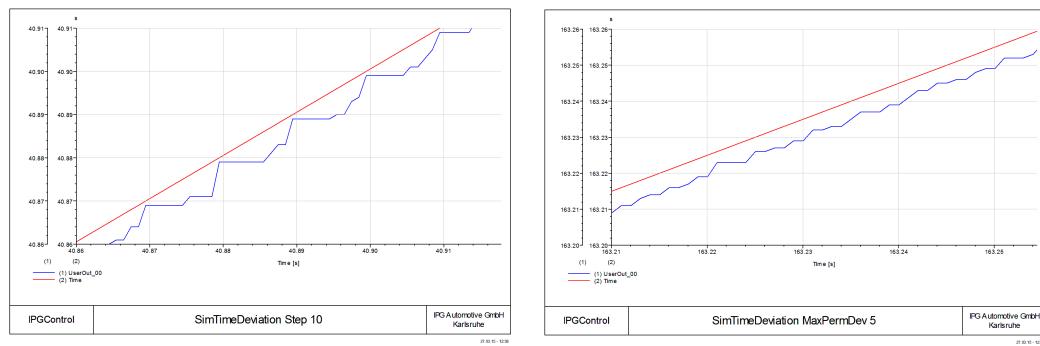


Figure 10.7: Comparison between option "Step = 10" (left) and "Max. Deviation = 5" (right). In the Step-Mode every 10th cycle the communication is synchronized. The "Max. Permissible Deviation"-Option results in more or less continuously delayed data.

## 10.2.4 Stream Settings

The second tab *Stream Settings* of the configuration dialog gives an overview of all configured data streams. It allows the user to choose a DDL file that describes the structure of the data streams exchanged with ADTF. It also enables the user to specify the streams that are send to or received from ADTF.

Adding another stream to a configuration can be achieved by clicking the button with the green plus symbol at the left bottom of the configuration dialog. In case an existing stream is selected before, a copy of this stream will be added. Accordingly a Stream can be removed by selecting it and pushing the red minus button.

For each stream element a mapping to a UAQ can be defined (see [section 'Variable Mapping'](#)). A possible configuration with three incoming and three outgoing streams is shown in [Figure 10.8](#).

For convenience it is possible to expand and collapse the visualization of all streams of a configuration with a click on the corresponding buttons. It is also possible to hide all variables that are not mapped to UAQs by activating "*Hide unmapped elements*".

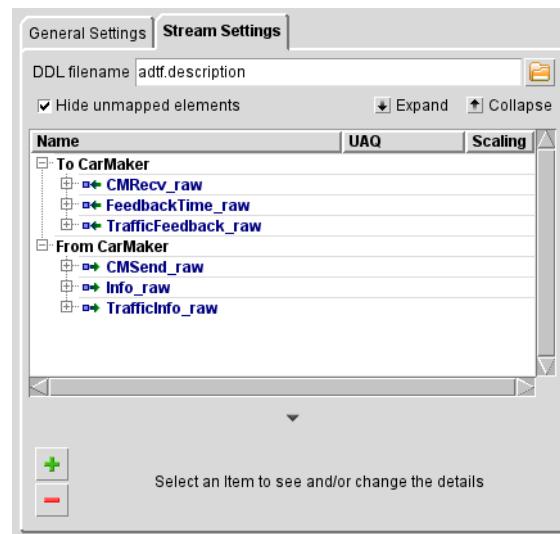


Figure 10.8: Stream Settings of the ADTF Interface

## DDL file

The description of the data streams by a DDL file is mandatory. Both the CarMaker user interface and the simulation environment read this data. The DDL file must be placed in the CarMaker project directory under *Data/Config*.

The ADTF interface of CarMaker uses the same DDL data as ADTF. The used library for reading xml files supports only utf-8 and utf-16 encoding. If special characters or umlauts are used in the DDL it is recommended to use utf-8 encoding.

The following is an excerpt from a DDL file. This section shows a Stream Definition:

```
1:  <streams>
2:    <stream name="myStruct_raw" type="adtf.core.media_type">
3:      <struct bytepos="0" type="tmyStruct_raw" />
4:    </stream>
5:  </streams>
6:
7:  <structs>
8:    <struct name="tmyStruct_raw" version="1">
9:      <element alignment="1" arraysize="1" byteorder="Intel" bytepos="0"
name="Counter" type="tUInt32" />
10:     <element alignment="1" arraysize="3" byteorder="Intel" bytepos="4" name="test"
type="taStruct1" />
11:   </struct>
12:   <struct name="taStruct1" version="1">
13:     <element alignment="1" arraysize="1" byteorder="Intel" bytepos="0" name="arr1"
type="tFloat32" />
14:     <element alignment="1" arraysize="2" byteorder="Intel" bytepos="4" name="arr2"
type="tUInt32" />
15:   </struct>
16: </structs>
```

It is possible to generate CarMaker UAQs for all stream elements. In case of automatic generation the designation of the UAQs is derived from the DDL description file: the designation structure is built from the element name, and begins with the stream name. If case arrays are used, a corresponding counter is added to the identifier. The example shown above leads to the following UAQs:

```
1: myStruct_raw.Counter
2: myStruct_raw.test.0.arr1
3: myStruct_raw.test.0.arr2.0
4: myStruct_raw.test.0.arr2.1
5: myStruct_raw.test.1.arr1
6: myStruct_raw.test.1.arr2.0
7: myStruct_raw.test.1.arr2.1
8: myStruct_raw.test.2.arr1
9: myStruct_raw.test.2.arr2.0
10: myStruct_raw.test.2.arr2.1
```



To activate automatic mapping of all elements, right-click on a stream and choose "*Map complete stream*", like shown in [Figure 10.9](#). This will create a UAQ for each stream element. Please note, that this option covers up existing mappings while it is activated. Also note that mapping of complete streams is not available for streams that contain dynamic arrays.

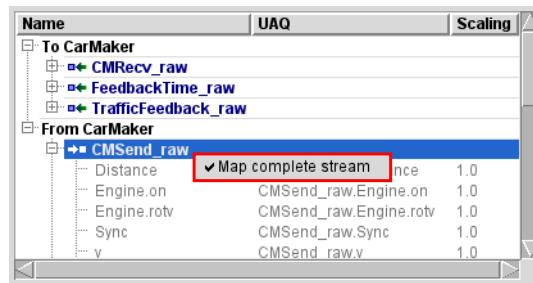


Figure 10.9: Right-click to activate mapping of complete stream

## Stream Info

After selecting a stream the section *Stream Info* becomes visible and the chosen data stream can be configured.

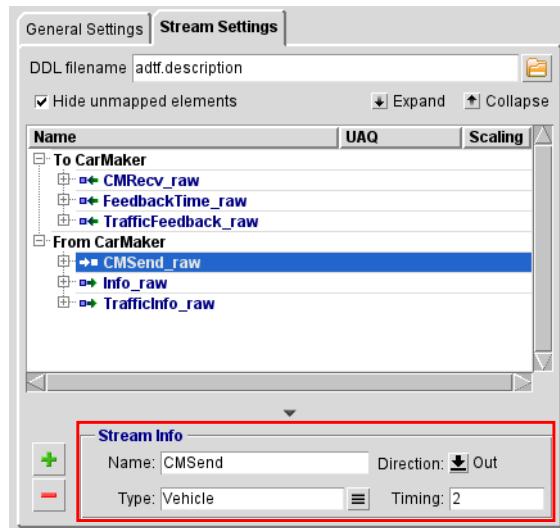


Figure 10.10: Configuration of a Data Stream

**Name** The name of the data stream. This name is used while assigning the data packets during the data exchange. The stream name must be unique. It may therefore be used only once in a configuration. The stream name corresponds to the port name in the ADTF environment. Since the CarMaker ADTF Interface uses ADTF-to-X communication, the mandatory suffix "\_raw" is automatically added to the stream name.

**Type** The type defines the data type of ADTF data stream (stream name in the DDL file). A selection of an available type of the current DDL file can be done by clicking on the options button next to the edit field. A stream type can be used several times.

**Direction** Using the drop-down menu, one can choose the direction of the data stream. "In" configures an incoming data stream sent by ADTF. "Out" is the configuration for sending a data stream.

**Timing** For the data streams to be sent a transmission cycle must be configured. This is specified in CarMaker cycles (1ms).

## Variable Mapping

The ADTF variables of the individual data streams can be mapped to CarMaker UAQs. A particular mapping can be edited by selecting a stream element as shown in [Figure 10.11](#).

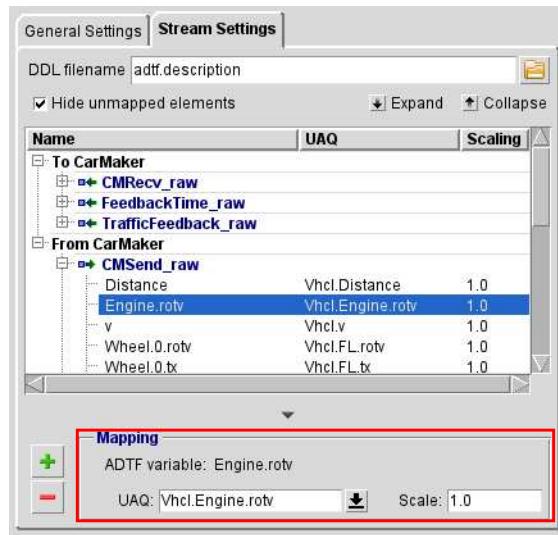


Figure 10.11: Mapping of a variable

- ADTF variable** Name of the selected ADTF variable to be mapped.
- UAQ** User Accessible Quantity from CarMaker data dictionary to be linked. Use the drop-down menu to choose between all available quantities.
- Scale** An optional scaling factor can be defined. The default value is 1.0. The respective calculations are:

$$ADTF = \frac{UAQ}{Scaling}$$

$$UAQ = ADTF \times Scaling$$



Please note, that element mapping to UAQs can only be performed for fixed size streams but not for streams containing dynamic arrays. Dynamic streams can be received and send via the C-Code API (see [section 10.2.7](#)).

### 10.2.5 Structure of the Configuration Infofile

All settings in the configuration dialog are written to an Infofile. The key names for the general settings (see [section 10.2.3 'General Settings'](#)) are as follows:

---

**ADTF.hostname = name**

---

Name of the ADTF computer system or IP-Address. Default: -.

---

**ADTF.ADTFPort = 12345**

---

UDP-Port on the ADTF system to send the data. Default: 5001.

---

**ADTF.CMPort= 12345**

---

UDP-Port through which the ADTF system receives the data. Default: 5000.

---

**ADTF.DDL\_filename = *name.dll***

---

Name of the DDL file which is expected in the *Data/Config* folder of the CarMaker project directory. Default: -.

---

**ADTF.cpuID = *ID***

---

Only for HIL: Send and Receive Task is running on the specified CPU (-1 = automatic configuration of the operating system). Default: -1.

---

**ADTF.Stream.In:**  
***signal name*      *stream name***

---

Two column Info-File-Text-Entry:

1. Name of the signals in ADTF
2. Name of the streams (= signal type in ADTF)

---

**ADTF.Stream.Out:**  
***signal name*      *stream name*      *ncycles***

---

Three column Info-File-Text-Entry:

1. Name of the signals in ADTF
2. Name of the streams (= signal type in ADTF)
3. Transmission rate in CarMaker cycles in ms (integers only)

The Infofile parameters for the synchronization between the CarMaker ADTF Interface and ADTF are the following (see [section 'Synchronization Options'](#) for details):

---

**ADTF.Sync.mode = *value***

---

Possible synchronization modes (default: 0):

0: "Off"  
1: "ADTF Time"  
2: "Incoming Stream"  
3: "In- and outgoing Stream"  
4: "Sync Signal"

---

**ADTF.Sync.StreamName = *string***

---

Name of an incoming stream.

---

**ADTF.Sync.StreamNameOut = *string***

---

Name of an outgoing stream.

---

**ADTF.Sync.ExpUpdCycleCount = *value***

---

Expected update cycle count. Default: 0.

---

**ADTF.Sync.sigNameIn = *string***

---

Name of data stream with variables for incoming synchronization counters.

---

**ADTF.Sync.sigNameOut = *string***

---

Name of data stream with variables for outgoing synchronization counters.

The signal specific Infofile parameters are as listed below:

---

**ADTF.<signalname>.showDDict = *bool***

---

If set to 1, DataDictionary entries are created for every element in the streams. The quantity name is derived from the structure of the stream. Default: 0.

---

**ADTF.<signalname>.StreamMap:**  
***name ADTF signal*      *name UAQ*      *factor***

---

Defines the mapping between stream elements and the UAQs. The configuration is done by a 3-column Infofile text entry:

- The first column consists of the ADTF name that is called. This refers to an element in the stream.

- The second column contains an UAQ, with which the ADTF variable has to be linked.
- The third column defines an optional scaling factor. The third column being left blank, would then assume a scaling factor of 1.0.

## 10.2.6 Example Configuration

This section represents a step-by-step instruction for data exchange with CarMaker on the ADTF Message Bus UDP Extension. This guide is based on ADTF version 2.7.2. There may be variations in case of other ADTF versions being used.

### 1. Creating a new Project

- Go to *File -> Create New Project...*
- Select the "empty" template
- Provide a project name - e.g. "MyCarMakerConnection"
- Select the project path
- Copy Existing ddl (e.g. adtf.description) in new project directory

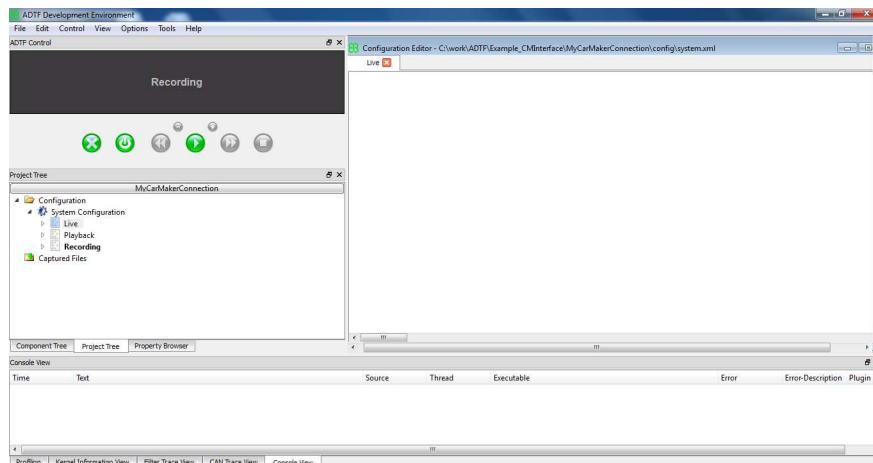


Figure 10.12: Newly created ADTF project

### 2. Adaption of the General Settings:

For this, go to *Edit -> Configuration -> Configuration Properties*

Property	Value
Media Description: Files	<Choose the ddl-file> (e.g. /adtf.description)
Message Bus: Enable message bus	True
Message Bus: Enable synchronizing	False
Session: Data exchange default local port	5000
Session: Data exchange default protocol	udp

Table 10.1: General Settings in ADTF to use the ADTF Message Bus

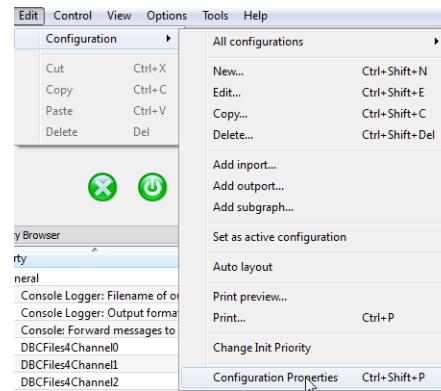


Figure 10.13: Open the General project configurations

### 3. Insert CarMaker configuration as "subgraph" in current configuration

- Create new configuration (representing CarMaker)  
*Edit -> Configuration -> New (Name: CarMaker)*
- Insert subgraph "CarMaker" for self-configuration
  - Select Custom Configuration
  - Add CarMaker subgraph (*Edit -> Configuration -> Add subgraph*, choose "CarMaker")
- Configure CarMaker "subgraph"
  - Select the subgraph
  - Edit the properties:

Property	Value
url	udp://<ip-address:port> (e.g. udp://192.168.0.113:5000)

Table 10.2: Settings of the CarMaker "Subgraphs"

- Select the CarMaker Configuration for the specification of data exchange
- Define the input and output ports:
  - Go to *Edit -> Configuration -> Add import... / Add output...*
  - Ports usage - To use the configuration of the ADTF-to-X-Communication the port name must end with \_raw
  - Stream Name (type of data packets) as a property of the port state (stream name is specified in the ddl and must be available there)

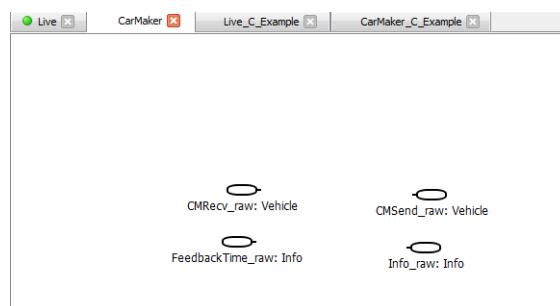


Figure 10.14: Subgraph CarMaker

- User Configuration (e.g. "Live") choice and Ports of CarMaker to be connected with "Subgraphs"

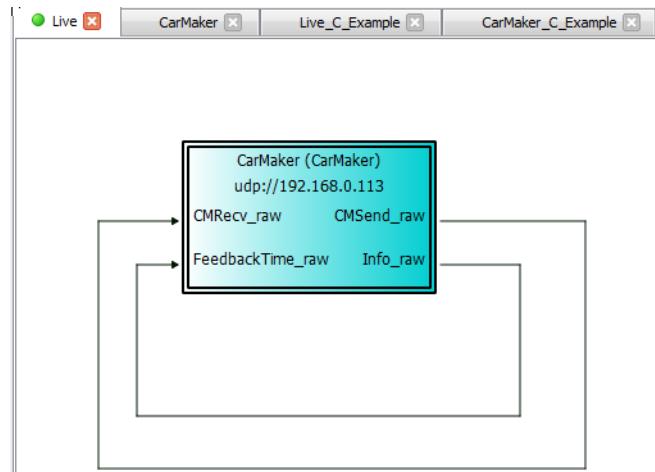


Figure 10.15: "Live"-Configuration with the interface to CarMaker (associated sub-graph)

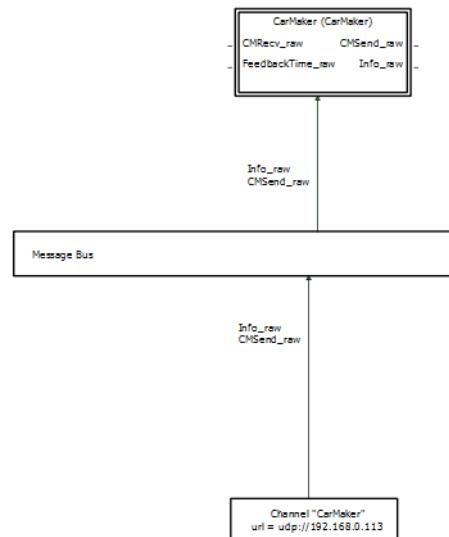


Figure 10.16: Overview of the network configuration created in ADTF ("Network Overview")

### 10.2.7 C-Code API

In addition to the parameter-based functions there are additional C-functions. Using these C functions, the interface with ADTF can be implemented without the Data Dictionary module in CarMaker. Data types that are not supported by the Data Dictionary can be processed. An example for this is calling a string in a character array.

#### Request for the address of an ADTF Stream

With the following functions, the pointer for the memory space the ADTF Stream is stored in can be requested:

---

## ADTF\_GetDataPtr ()

---

```
int ADTF_GetDataPtr (char *name, void **ptr)
```

### Arguments

*Name*: Name of the ADTF-Stream or Stream element.

*Ptr*: Address of a pointer which must be defined by the user. The ADTF interface stores where the address of the memory is. *ptr* is set to NULL if the name is not a valid identifier.

### Return Value

After a successful call, the size of the data is returned in bytes. Otherwise, the return value is negative.

## Synchronizing the Local Storage area

The following functions associate a global memory space to the data streams that are sent or received. Before sending the data is copied into the output buffer of the data stream or cyclically copied from the input buffer to the defined memory area.

---

## ADTF\_RequestData ()

---

```
int ADTF_RequestData (char *name, void * localData, int size, char dir)
```

### Arguments

*Name*: Name of the ADTF pins (e.g. ToCM\_raw) or name of a data area in the stream.

*localData*: The data area to be synchronized with ADTF.

*size*: Size in bytes of the data to be copied.

*dir*: Synchronization direction of data - sending 's' or receive 'r'.

### Return Value

After a successful call, the job ID of the synchronization requested is returned. Otherwise, the return value is negative.

## Removing the Storage Area Synchronization

The function to synchronize a memory space can be removed again.

---

## ADTF\_DeleteDataRequest () ADTF\_DeleteDataRequest\_byName ()

---

```
int ADTF_DeleteDataRequest (int id)
int ADTF_DeleteDataRequest_byName (char *name)
```

## Arguments

*id*: Job ID of the synchronization previously requested (= return value from ADTF\_RequestData). Alternatively, the name of linked ADTF pins or designation of the synchronized array in the stream (analogous to the previously made statement in ADTF\_RequestData) can be used.

## Return Value

A successful call returns 0, while an unsuccessful one returns a negative value.

## Interface Function to Send

A hook function allows custom functions to be executed every time an ADTF stream is sent. Considering different functions are executed with respect to different streams, the stream name will be communicated.

A negative return value prevents the sending of ADTF streams. The return value should be 0 or greater than that for the stream to be sent.

Example code for a user interface which can be found in the User.c file:

```
1: static int
2: User_ADTF_TxHook (char* streamName) {
3:
4:     if (strcmp(streamName, "CMSSend_raw") == 0) {
5:         /* do something... */
6:     }
7:
8:     return 0;
9: }
```

To be activated, the function needs to be called in the *User\_Init()* part of the User.c file:

```
1: int
2: User_Init (void)
3: {
4: ...
5:     ADTF_TxHookFct = User_ADTF_TxHook;
6: ...
7: }
```

## 10.3 Powertrain: CarMaker - CRUISE Interface

The simulation tool CRUISE is developed by the Advanced Simulation Techniques (AST) - team of the AVL List GmbH. CRUISE is a software for simulating driving performance, fuel consumption, emissions and driving quality. Although CRUISE is able to simulate the whole vehicle, the focus is on the simulation of the powertrain.

The CarMaker-CRUISE Interface is based on a co-simulation of CarMaker and CRUISE. If a CRUISE powertrain is chosen in the *Vehicle Data Set*, CarMaker starts the CRUISE simulation program during the initialization phase of the TestRun. During the simulation the CRUISE powertrain receives the driver's inputs and the current vehicle state from CarMaker and feeds back the current powertrain state (e.g. wheel speed, engine speed) to CarMaker.

Even though the CRUISE frontend is only available for Windows systems the simulation core is also available for linux systems. Therefore the CRUISE powertrain can be used with the Xpack4. The CRUISE interface currently is available for CarMaker and TruckMaker only.

### 10.3.1 Setting up the CarMaker - CRUISE Interface

There are two ways to prepare a CarMaker project for the co-simulation with AVL CRUISE:

- copy the CRUISE library to the folder *lib* inside the CarMaker project directory
- set the path to the CRUISE installation inside the *SimParameters* file of the CarMaker project using the key *Cruise.LibraryPath*

There are some optional settings for CRUISE that can be made inside *SimParameters*. For further information about these special settings, see [section "SimParameters" in the Reference Manual](#).

### 10.3.2 Simulating with a CRUISE Powertrain

Assuming you have a suitable CRUISE powertrain model it is quite easy to integrate this model into CarMaker. In the *Vehicle Data Set* tab *Powertrain* select *Powertrain Model: AVL CRUISE* and the interface shown in [Figure 10.17](#) appears.

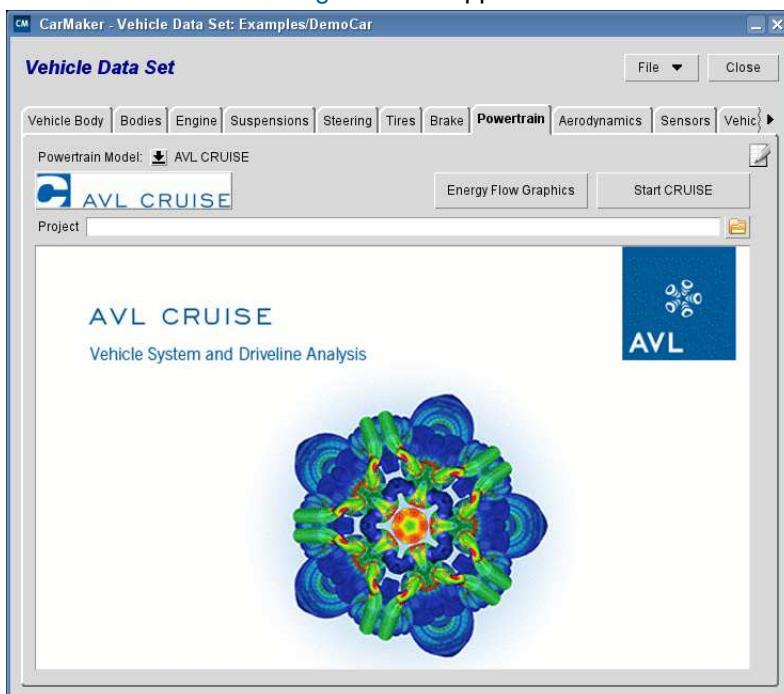


Figure 10.17: Vehicle Data Set: Powertrain folder with Powertrain-Kind AVL\_CRUISE

**Project** Here, the path to the actual CRUISE project is shown. If the field is empty, no valid CRUISE project is selected. By clicking on the file icon at the right a file browser pops up. Valid CRUISE projects with the file extension .prj can be chosen. If there is a picture of the powertrain model in the CRUISE project folder, it will be displayed in the area at the bottom as shown in [Figure 10.18](#). See the next chapter for the requirements a CRUISE project has to comply.

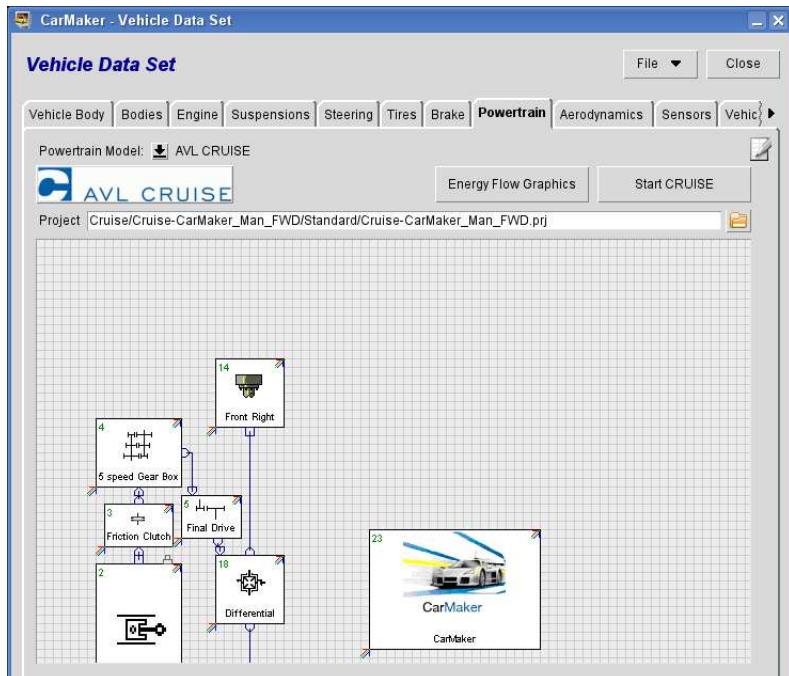


Figure 10.18: CRUISE powertrain model view

As soon as the start button of CarMaker is pressed, CRUISE will be started automatically with the selected project.

**AVL CRUISE** The left button with the CRUISE icon will update the CRUISE powertrain model picture shown in the big area at the bottom of the folder. If no valid model is chosen or no .gif file of the model exists, a CRUISE default picture is shown.

**Energy Flow Graphics** This button will start the CRUISE energy flow graphics program (available for Windows only). To start its animation, choose the application and then press the connect button.

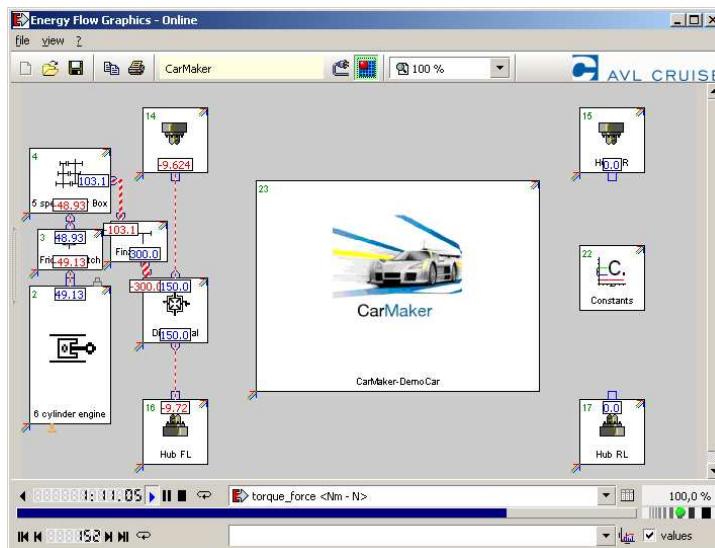


Figure 10.19: CRUISE: Energy Flow Graphics

**Start CRUISE** The right button will start CRUISE and if possible load the currently chosen powertrain project (available for Windows only).



It is not mandatory to start CRUISE before pressing CarMaker's start button - this will be done automatically.

### 10.3.3 Preparing the CRUISE Powertrain Model

On CarMaker side, the vehicle's powertrain model has to be set to AVL CRUISE and a CRUISE project has to be selected as described in the previous chapter to establish the co-simulation. On CRUISE side, there are special CarMaker modules *CM-Car* and *Hub* to enable the co-simulation. Please take care that the following requirements are complied by the CRUISE project that you want to use with CarMaker:

- there must be exactly one *CM-Car* module in the model
  - mind to select the correct module for CarMaker 5.0 (for older CarMaker versions refer to old CarMaker documentation)
- there have to be exactly four *Hub* modules (make sure there is one for each wheel)
- SAM-Task must be activated and the value for *Time Step* has to be set to 0.001s
- there must not be any *Vehicle* or *Cockpit* modules
- please make sure that the engine module gets a plausible signal for channel *Start Switch*

Please also consult the CRUISE *Cruise-CarMaker Interface* manual about how to built up your own CarMaker suitable powertrain model.

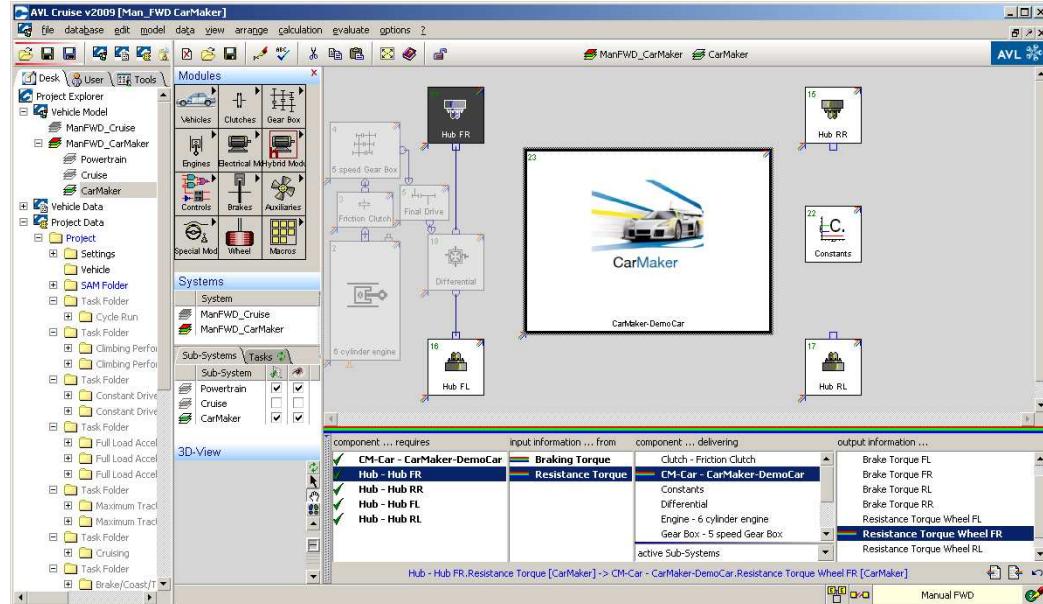


Figure 10.20: CRUISE powertrain project with CarMaker interface modules

## CM-Car Module

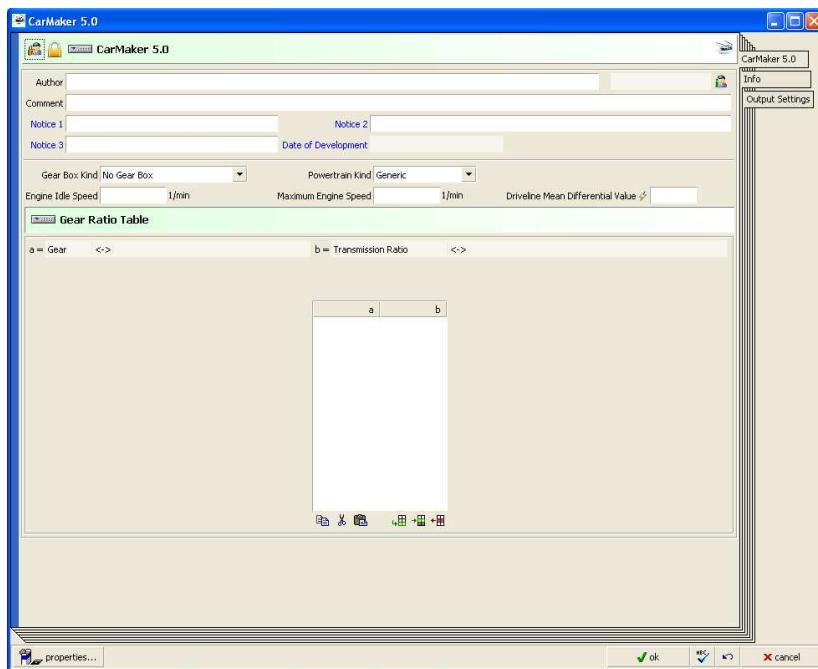


Figure 10.21: CRUISE: CM-Car module GUI

There are only a few parameters which have to be set inside the *CM-Car* module:

- *Gear Box Kind* tells IPGDriver about the kind (manual, automatic, none) of gearbox used in the powertrain. Manual gearboxes are shifted by IPGDriver, automatic gearbox have to be handled inside the CRUISE model.
- *Powertrain Type* defines the powertrain architecture (e.g. conventional, electric, axle split hybrid)
- *Engine Idle Speed* gives the idle speed of the engine. Make sure that the value is consistent with the engine idle speed inside the CRUISE engine model
- *Maximum Engine Speed* gives the maximum rotation speed of the engine. Make sure that the value is consistent with the maximum engine speed inside the CRUISE engine model.
- *Driveline Mean Differential Value* provides the transmission between gearbox output speed and wheel speed. This information is required by IPGDriver.
- *Gear Ratio Table*, this table contains the ratios of the various gears of the model.

Please refer to [section 10.3.4 'CM-Car Interface IO-Signals'](#) for more information about the CM-Car module's interface signals.

**Hub Module** In the *Hub* module (see [Figure 10.22: CRUISE: Hub module GU](#)) there are even less parameters to set. Although it is possible to set a value for the *Inertia Moment*, this is not necessary because this value is overwritten with the wheel inertia given in the *Vehicle Data Set* of CarMaker.

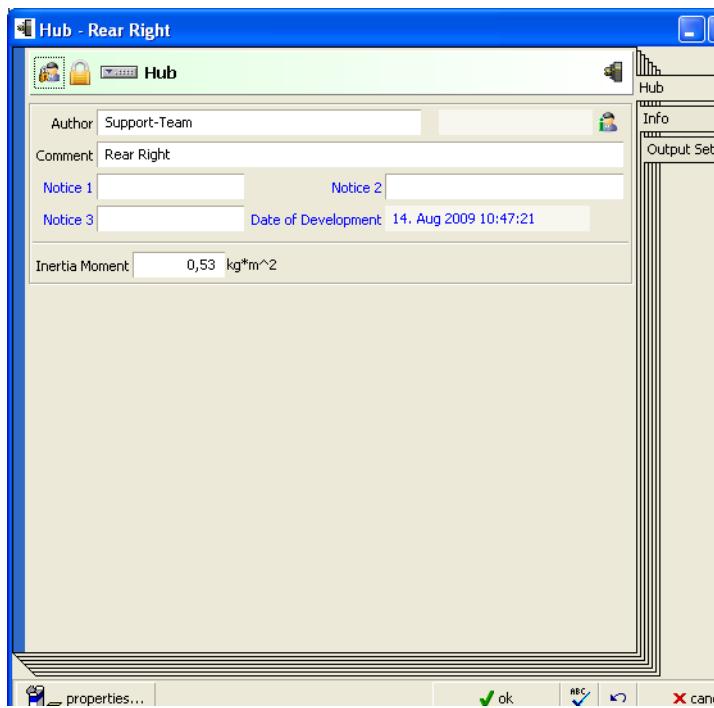


Figure 10.22: CRUISE: Hub module GU



It is mandatory that there is exactly one hub module for each wheel (front left, front right, rear left, rear right). To change the location of a wheel hub, open the *Properties* dialog and choose the wanted location in the pull down menu.

To avoid problems with slightly rotating wheels during standstill, it is recommended to uncheck the *Dynamic Mode* as shown in [Figure 10.23](#).

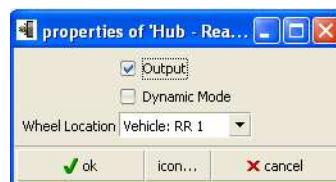


Figure 10.23: CRUISE: Hub module properties

#### Cruise Interface Module

This module is optional. It allows to modify signals using the Direct Variable Access (DVA) functionality of CarMaker.

To use this feature, the CRUISE project has to contain a *Cruise Interface* module. The *Interface Label* entry of this module must be *CM-DVA*. For all connected signals in the CRUISE project, User Accessible Quantities will be created automatically. CarMaker does not know anything about the unit of these signals, but the values of these signals refer to SI-Units (e.g. rad/s for rotation speeds). The notation for signals from CarMaker to CRUISE is as follows: *AVL\_CRUISE.DVA.SI.out<ChannelNumber>.<CruiseSignalName>* and for signals coming from CRUISE to CarMaker: *AVL\_CRUISE.DVA.SI.in<ChannelNumber>.<CruiseSignalName>*.

If the following key `Cruise.CM-DVA.SkipQuants = 0` is set in the SimParameters file, additional quantities with the CRUISE signal names are created. Unlike the DVA quantities mentioned before, the unit defined in the CM-DVA module is attributed to these quantities. So it is in the users responsibility to take care about unit conversion, if they are used for further calculations inside CarMaker. Notation for these quantities is as follows:

`AVL_CRUISE.DVA.out<ChannelNumber>.<CruiseSignalName>`

`AVL_CRUISE.DVA.in<ChannelNumber>.<CruiseSignalName>`

These values are thought to be read only, trying to sent these quantity values from CarMaker to CRUISE. e.g. using a *Write CM Dict* block in CM4SL will have no effect to the Cruise signal.

### 10.3.4 CM-Car Interface IO-Signals

The CM-Car module offers a prewired set of interface signals that can be found in [Table 10.3: Signals from CarMaker to CRUISE](#) and [Table 10.4: Signals from CRUISE to CarMaker](#). If these signals are connected inside the CRUISE powertrain model, CarMaker automatically creates User Accessible Quantities which include the CRUISE signal names. The CRUISE quantities are found under the prefix `AVL_CRUISE`, e.g.: `AVL_CRUISE.Car.in00.Wheel_FL_Angle`.

If the CRUISE project has been exchanged or modified during the current CarMaker session, it is necessary to restart the application to get the quantities updated in Data Dictionary.

Additionally, It is possible to implement further freely definable interfaces by using the Car-Maker ModelManager. Please refer to the Programmer's Guide for further information and have a look to the Example 'MyCruiself.c' in your CarMaker installation directory under `hil/<CarMaker Version>/Templates/Car_Extras/src/ExtraModels/`

### Interface Outports: CarMaker -> CRUISE

Table 10.3: Signals from CarMaker to CRUISE

Channel	CarMaker Variable	CRUISE Signal Name	Unit	Required
0	SimCore.DeltaT	Simulation Step Size	s	no
1	SimCore.Time	Simulation Time	s	no
2	Vehicle.Pol_Vel_1[0]	Car Velocity	m/s	no
3	VehicleControl.Brake	Brake Signal	-	(yes) <sup>a</sup>
4	VehicleControl.Clutch	Clutch Release	-	no
5	VehicleControl.Gas	Load Signal	-	yes
6	VehicleControl.GearNo	Desired Gear	-	no
7	Car.Tire[0].Trq_T2W	Resistance Torque Wheel FL	Nm	yes
8	Car.Tire[1].Trq_T2W	Resistance Torque Wheel FR	Nm	yes
9	Car.Tire[2].Trq_T2W	Resistance Torque Wheel RL	Nm	yes
10	Car.Tire[3].Trq_T2W	Resistance Torque Wheel RR	Nm	yes
11	Car.ConBdy1.atHori	Acceleration	m/s <sup>2</sup>	no
12	Car.ConBdy1.alHori	Acceleration Lateral	m/s <sup>2</sup>	no
13	Vehicle.LongSlipFL	Longitudinal Slip FL	-	no
14	Vehicle.LongSlipFR	Longitudinal Slip FR	-	no
15	Vehicle.LongSlipRL <sup>b</sup>	Longitudinal Slip RL	-	no
16	Vehicle.LongSlipRR <sup>c</sup>	Longitudinal Slip RR	-	no
18	VehicleControl.Key	Key	-	no
19	VehicleControl.SST	SST	-	no
20	Brake.Trq_tot[0]	Brake Torque FL	Nm	(yes)
21	Brake.Trq_tot[1]	Brake Torque FR	Nm	(yes)
22	Brake.Trq_tot[2]	Brake Torque RL	Nm	(yes)
23	Brake.Trq_tot[3]	Brake Torque RR	Nm	(yes)
50	Brake.IF.Trq_Reg_trg[0]	Target Powertrain Brake Torque FL	Nm	(yes)
51	Brake.IF.Trq_Reg_trg[1]	Target Powertrain Brake Torque FR	Nm	(yes)
52	Brake.IF.Trq_Reg_trg[2]	Target Powertrain Brake Torque RL	Nm	(yes)
53	Brake.IF.Trq_Reg_trg[3]	Target Powertrain Brake Torque RR	Nm	(yes)
60	VehicleControl.SelectorCtrl	Selector Control	-	no
61	PowerTrain.ControlIF.StrategyMode_trg.SetManual	Activate Manual Strategy Mode Target	-	no
62	PowerTrain.ControlIF.StrategyMode_trg	Strategy Mode Target	-	no
63	PT.Control.Consump.Reset	Consumption Reset	-	no
70	VehicleControl.UserSignal[0]	User Signal 0	-	no

Table 10.3: Signals from CarMaker to CRUISE

Channel	CarMaker Variable	CRUISE Signal Name	Unit	Required
71	VehicleControl.UserSignal[1]	User Signal 1	-	no
72	VehicleControl.UserSignal[2]	User Signal 2	-	no
73	VehicleControl.UserSignal[3]	User Signal 3	-	no
74	VehicleControl.UserSignal[4]	User Signal 4	-	no

a. whether the brake signal or the brake torques (channel 20..23) should be connected,

Note: no error will be generated if neither of them is connected

b. in case of twin tires (Vehicle.LongSlipRL + Vehicle.LongSlipTwinRL) / 2.0

c. in case of twin tires (Vehicle.LongSlipRR + Vehicle.LongSlipTwinRR) / 2.0

### Interface Imports: CRUISE -> CarMaker

Table 10.4: Signals from CRUISE to CarMaker

Channel	CRUISE Signal Name	CarMaker Variables	Unit	Required
0	Wheel FL Angle	PowerTrain.WFL.rot	rad	yes <sup>a</sup>
1	Wheel FR Angle	PowerTrain.WFR.rot	rsd	yes
2	Wheel RL Angle	PowerTrain.WRL.rot	rad	yes
3	Wheel RR Angle	PowerTrain.WRR.rot	rad	yes
4	Wheel FL Speed	PowerTrain.WFL.rotv	rad/s	yes
5	Wheel FR Speed	PowerTrain.WFR.rotv	rad/s	yes
6	Wheel RL Speed	PowerTrain.WRL.rotv	rad/s	yes
7	Wheel RR Speed	PowerTrain.WRR.rotv	rad/s	yes
8	Ignition <sup>b</sup>	PT.Control.Ingnition	-	(yes)
9	Engine Speed	PowerTrain.Engine.rotv	rad/s	no
10	Engine Torque	PowerTrain.Engine.Trq	Nm	no
11	Operation State	PowerTrain.ControlIF.OperationState	-	(yes)
12	Operation Error	PowerTrain.ControlIF.OperationError	-	no
13	Strategy Mode	PowerTrain.ControlIF.StrategyMode	-	(yes)
14	GB GearNo	PowerTrain.GearBox.GearNo	-	no
15	Supp Brake Torque FL	PowerTrain.WFL.Trq_B2W	Nm	no
16	Supp Brake Torque FR	PowerTrain.WFR.Trq_B2W	Nm	no
17	Supp Brake Torque RL	PowerTrain.WRL.Trq_B2W	Nm	no
18	Supp Brake Torque RR	PowerTrain.WRR.Trq_B2W	Nm	no
19	Drive Torque FL	PowerTrain.WFL.Trq_DL2W	Nm	no
20	Drive Torque FR	PowerTrain.WFR.Trq_DL2W	Nm	no
21	Drive Torque RL	PowerTrain.WRL.Trq_DL2W	Nm	no
22	Drive Torque RR	PowerTrain.WRR.Trq_DL2W	Nm	no
23	Front Supp2Bdy Torque X-axis	PowerTrain.Trq_Supp2Bdy1[0]	Nm	no
24	Front Supp2Bdy Torque Y-axis	PowerTrain.Trq_Supp2Bdy1[1]	Nm	no
25	Front Supp2Bdy Torque Z-axis	PowerTrain.Trq_Supp2Bdy1[2]	Nm	no

Table 10.4: Signals from CRUISE to CarMaker

Channel	CRUISE Signal Name	CarMaker Variables	Unit	Required
26	Rear Supp2Bdy Torque X-axis	PowerTrain.Trq_Supp2Bdy1B[0]	Nm	no
27	Rear Supp2Bdy Torque Y-axis	PowerTrain.Trq_Supp2Bdy1B[1]	Nm	no
28	Rear Supp2Bdy Torque Z-axis	PowerTrain.Trq_Supp2Bdy1B[2]	Nm	no
29	Supp Torque Engine x-Axis	PowerTrain.Trq_Supp2BdyEng[0]	Nm	no
30	Supp Torque Engine y-Axis	PowerTrain.Trq_Supp2BdyEng[1]	Nm	no
60	Wheel Carrier Supp Torque (Y.axis) FL	PowerTrain.IF.WheelOut[FL].Trq_Supp2WC	Nm	(yes)
61	Wheel Carrier Supp Torque (Y.axis) FR	PowerTrain.IF.WheelOut[FR].Trq_Supp2WC	Nm	(yes)
62	Wheel Carrier Supp Torque (Y.axis) RL	PowerTrain.IF.WheelOut[RL].Trq_Supp2WC	Nm	(yes)
63	Wheel Carrier Supp Torque (Y.axis) RR	PowerTrain.IF.WheelOut[RR].Trq_Supp2WC	Nm	(yes)
70	Reg Brake Torque FL	PowerTrain.IF.WheelOut[FL].Trq_BrakeReg	Nm	(yes)
71	Reg Brake Torque FR	PowerTrain.IF.WheelOut[FR].Trq_BrakeReg	Nm	(yes)
72	Reg Brake Torque RL	PowerTrain.IF.WheelOut[RL].Trq_BrakeReg	Nm	(yes)
73	Reg Brake Torque RR	PowerTrain.IF.WheelOut[RR].Trq_BrakeReg	Nm	(yes)
80	Max Reg Brake Torque FL	PowerTrain.IF.WheelOut[FL].Trq_BrakeReg_max	Nm	(yes)
81	Max Reg Brake Torque FR	PowerTrain.IF.WheelOut[FR].Trq_BrakeReg_max	Nm	(yes)
82	Max Reg Brake Torque RL	PowerTrain.IF.WheelOut[RL].Trq_BrakeReg_max	Nm	(yes)
83	Max Reg Brake Torque RR	PowerTrain.IF.WheelOut[RR].Trq_BrakeReg_max	Nm	(yes)
90	Actual Fluel Consump-tion	PT.Control.Consump.Fuel.Act	l	no
91	Average Fluel Con-sumption	PT.Control.Consump.Fuel.Avg	l/ 100km	no
92	Absolute Fluel Con-sumption	PT.Control.Consump.Fuel.Abs	l	no
93	Actual Electric Con-sumption	PT.Control.Consump.Elec.Act	kWh	no
94	Average Electric Con-sumption	PT.Control.Consump.Elec.Avg	kWh/ 100km	no
95	Absolute Electric Con-sumption	PT.Control.Consump.Elec.Abs	kWh	no
96	Power Exploited	PT.Control.PwrExploited	%	no

Table 10.4: Signals from CRUISE to CarMaker

Channel	CRUISE Signal Name	CarMaker Variables	Unit	Required
97	Fuel Level	PowerTrain.Engine.Fuel.Level	%	(yes)
98	State of Charge	PowerTrain.BatteryCU_IF. SOC_HV/LV <sup>c</sup>	%	(yes)

a. channels 0..7 are connecting automatically with the corresponding hub modules

b. it is recommended to connect this signal to a value unequal zero

c. Low Volt (LV) when PT.Kind=Generic, otherwise High Volt (HV)

## 10.4 Powertrain: CarMaker - GT-SUITE Interface

The simulation tool GT-SUITE is developed by Gamma Technologies, Inc. It is a software for multi-physics simulation on divers levels of detail. Although GT-SUITE is able to simulate the whole vehicle, it is widely used as specialized tool for powertrain modeling and simulation.

Actually the CarMaker-GT-SUITE Interface is based on a co-simulation of CarMaker and GT-SUITE. If a GT-SUITE powertrain is chosen in the Vehicle Data Set CarMaker automatically starts the GT-SUITE simulation program during the initialization phase of the TestRun simulation. During the simulation the GT-SUITE powertrain receives the driver's inputs and the current vehicle state from CarMaker and feeds back the current powertrain state (e.g. wheel speed, engine speed) to CarMaker.

The GT-SUITE Interface currently is available for CarMaker and TruckMaker office versions and CarMaker/HIL Xeno using the GT-SuiteRT solver.

### 10.4.1 Setting up the CarMaker - GT-SUITE Interface

There are three ways to prepare a CarMaker project for the co-simulation with GT-SUITE:

- by default, CarMaker will look for the environment variable GTIHOME and if found, will use the suitable library all by itself. For using the GT-SuiteRT solver a installation of GT-SUITE is not mandatory, if GT-SUITE ist not installed use one of the following ways
- copy GT-SUITE library to the folder */lib* inside the CarMaker project directory
- set the path to the GT-SUITE library inside the *SimParameters* file of the CarMaker project using the key *GT.Lib*

Special options that should be used when starting GT-SUITE can be set in the Vehicle Data Set in the tab *Misc* as *Additional Parameters* using the key *PowerTrain.GT.GTLinkOption*.

There are some optional settings for GT-SUITE that can be made inside *SimParameters*. For further information about these special settings, see [section “SimParameters” in the Reference Manual](#).

## 10.4.2 Simulating with a GT-SUITE Powertrain

Assuming you have a suitable GT-SUITE powertrain model it is quite easy to integrate this model into CarMaker. In the *Vehicle Data Set* tab *Powertrain* select *Powertrain Model: GT-SUITE* and the interface shown in [Figure 10.24](#) appears.

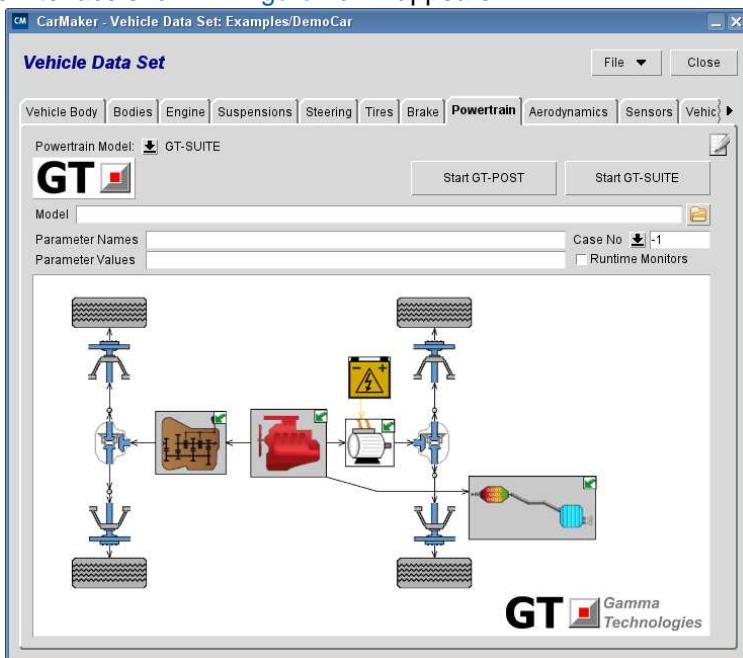


Figure 10.24: Vehicle Data Set: Powertrain folder with Powertrain-Kind GT-SUITE

**Model** Here, the path to the actual GT-SUITE model is shown. If the field is empty, no valid GT-SUITE project is selected. By clicking on the file icon at the right a file browser pops up. Valid GT-SUITE projects with the file extension .gmt or .dat can be chosen. For the realtime solver a .dat file has to be used. See the next chapter for the requirements a GT-SUITE model has to comply.

As soon as the start button of CarMaker is pressed, GT-SUITE will be started automatically with the selected model.

**Case No** This parameter defines the powertrain model's parameterization case that should be used, if several parameter cases are defined inside a .gmt file. "-1" means that the first checked case will be used.

**Parameter Names** In this field the powertrain model parameters that should be modified, can be defined. Insert the parameter names separated by space characters.

**Parameter Values** For each parameter inside *Parameter Names* a new value (a number or a string) has to be defined. The different parameter values are separated by space characters.

**Runtime Monitors** If *Runtime Monitors* is selected, all Monitors defined inside the GT-SUITE model will pop up at simulation start. Otherwise they will not be shown.

**GT** The left button with the GT-SUITE icon will update the GT-SUITE powertrain model picture shown in the big area at the bottom of the folder. If no valid model is chosen or no .gif file of the model exists, a GT-SUITE default picture is shown.

**Start GT-POST** This button will start the GT-POST programm.

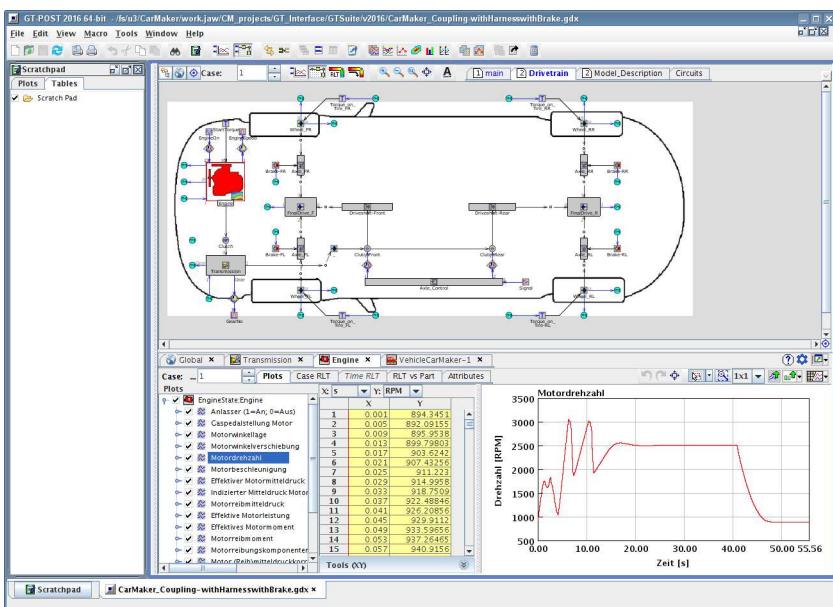


Figure 10.25: GT-POST

**Start GT-SUITE** The right button will start GT-SUITE and if possible load the currently chosen powertrain project.



It is not mandatory to start GT-SUITE before pressing CarMaker's start button - this will be done automatically.

### 10.4.3 Preparing the GT-SUITE Powertrain Model

On CarMaker side, the vehicle's powertrain model has to be set to GT-SUITE and a GT-SUITE project has to be selected as described in the previous chapter to establish the co-simulation. On GT-SUITE side, there is a special template *VehicleCarMaker* to enable the co-simulation. Please take care that the following requirements are complied by the GT-SUITE project that you want to use with CarMaker:

- there must be exactly one *VehicleCarMaker* template in the model

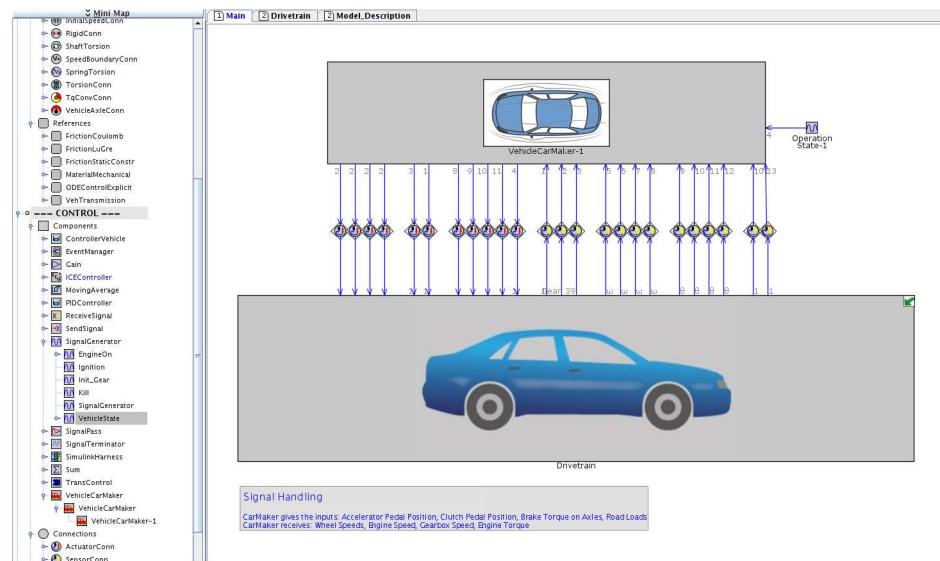


Figure 10.26: GT-SUITE powertrain project with CarMaker interface parts

### VehicleCarMaker

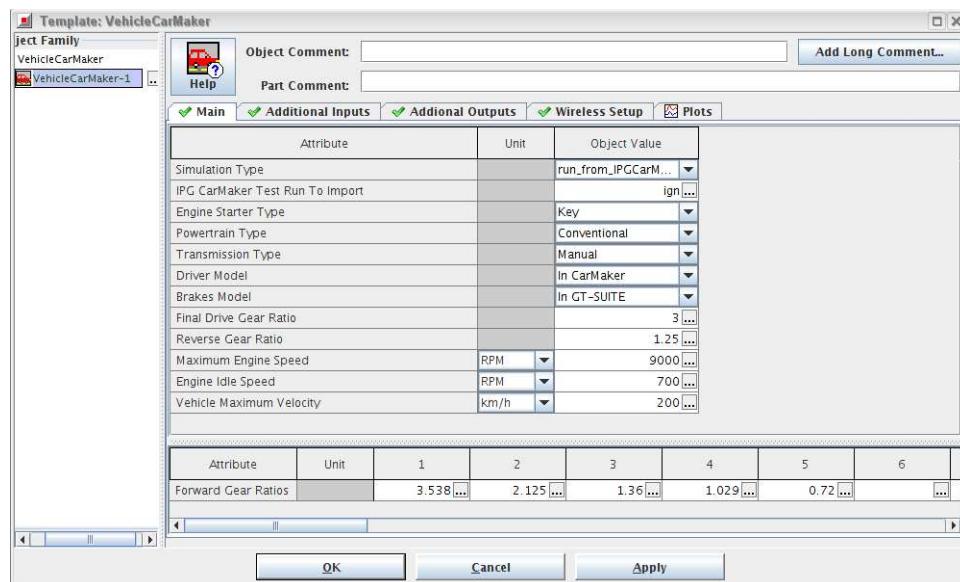


Figure 10.27: GT-SUITE: VehcileCarMaker part

There some parameters which have to be set inside the *VehicleCarMaker* template:

- *Simulation Type* specifies if the simulation is started by GT-SUITE or CarMaker. For the approach described here, please select *run\_from\_IPGCarMaker*.
- *IPG CarMaker TestRun To Import* is needed only if GT-SUITE starts the simulation. It defines the TestRun that should be used at CarMaker's side.
- *Driver Model* specifies if the CarMaker driver (recommended) or if a driver model is included in the GT-SUITE model. If the driver model is included in the GT-SUITE model then the brake model must also be included in the GT-SUITE mode
- *Brake Model* specifies if the CarMaker brake model is used or if the brake is included in the GT-SUITE model.

- *Powertrain Starter Type* tells CarMaker's maneuver control, if the powertrain is started with an ignition key or a start-stop-button
- *Powertrain Type* defines the powertrain architecture (e.g. conventional, electric, axle split hybrid)
- *Transmission Type* tells IPGDriver about the kind (manual, automatic, none) of gearbox used in the powertrain. Manual gearboxes are shifted by IPGDriver, automatic gearbox have to be handled inside the GT-SUITE model.
- *Final Drive Gear Ratio* provides the transmission between gearbox output speed and wheel speed. This information is required by IPGDriver.
- *Reverse Gear Ratio* corresponds to the reverse gear's transmission ratio
- *Maximum Engine Speed* gives the maximum rotation speed of the engine. Make sure that the value is consistent with the maximum engine speed inside the GT-SUITE engine model.
- *Engine Idle Speed* gives the idle speed of the engine. Make sure that the value is consistent with the engine idle speed inside the GT-SUITE engine model
- *Vehicle Maximum Velocity* provides the maximum possible vehicle speed for visualization in CarMaker's Instruments
- *Forward Gear Ratios*, this table contains the ratios of the various forward gears of the model



The wheel inertias will not automatically be set to the values provided in CarMaker. Please make sure that they are equal in GT-SUITE and CarMaker.

Please refer to [section 10.4.4 'VehicleCarMaker Interface IO-Signals'](#) for more information about the *VehicleCarMaker* module's interface signals.

#### 10.4.4 VehicleCarMaker Interface IO-Signals

The *VehicleCarMaker* module offers a prewired set of interface signals that can be found in [Table 10.5: Signals from CarMaker to GT-SUITE](#) and [Table 10.6: Signals from GT-SUITE to CarMaker](#). To access additional signals from CarMaker in GT-SUITE, just use the name of the corresponding User Accessible Quantity. New User Accessible Quantities will be created automatically for additional in- and outputs only. The naming convention is

*GT.Input.<signalID>.<signalname>*

for signals from GT-SUITE to CarMaker and

*GT.Output.<signalID>.<signalname>*

for signals from CarMaker to GT-SUITE respectively.

**Interface Outports: CarMaker -> GT-SUITE**

Table 10.5: Signals from CarMaker to GT-SUITE

Link ID	CarMaker Variable	GT-SUITE Signal Name	Unit	Required
1	VehicleControl.Gas	Accelerator Pedal	-	no
2	VehicleControl.Brake	Brake Pedal	-	no
3	VehicleControl.Clutch	Clutch Pedal	-	no
4	VehicleControl.GearNo	Desired Gear Number	-	no
5	VehicleControl.SelectorCtrl	Auto Gear Selector	-	no
6	VehicleControl.Key	Key Position	-	no
7	VehicleControl.SST	Start-Stop Button	-	no
8	Vehicle.FL.Trq_T2W	Front Left Wheel Torque	Nm	yes
9	Vehicle.FR.Trq_T2W	Front Right Wheel Torque	Nm	yes
10	Vehicle.RL.Trq_T2W	Rear Left Wheel Torque	Nm	yes
11	Vehicle.RR.Trq_T2W	Rear Right Wheel Torque	Nm	yes
12	Brake.Trq_tot[0]	Front Left Wheel Braking Torque	Nm	no
13	Brake.Trq_tot[1]	Front Right Wheel Braking Torque	Nm	no
14	Brake.Trq_tot[2]	Rear Left Wheel Braking Torque	Nm	no
15	Brake.Trq_tot[3]	Rear Right Wheel Braking Torque	Nm	no
16	Brake.IF.Trq_Reg_trg[0]	Front Left Wheel Target Regenerative braking Torque	Nm	no
17	Brake.IF.Trq_Reg_trg[1]	Front Right Wheel Target Regenerative braking Torque	Nm	no
18	Brake.IF.Trq_Reg_trg[2]	Rear Left Wheel Target Regenerative braking Torque	Nm	no
19	Brake.IF.Trq_Reg_trg[3]	Rear Right Wheel Target Regenerative braking Torque	Nm	no
20	Vehicle.FL.Trq_WhlBearing	Front Left Wheel Bearing Torque	Nm	no
21	Vehicle.FR.Trq_WhlBearing	Front Right Wheel Bearing Torque	Nm	no
22	Vehicle.RL.Trq_WhlBearing	Rear Left Wheel Bearing Torque	Nm	no
23	Vehicle.RR.Trq_WhlBearing	Rear Right Wheel Bearing Torque	Nm	no
56	Vehicle.FL2.Trq_T2W	Second Front Left Wheel Torque	Nm	no
57	Vehicle.FR2.Trq_T2W	Second Front Right Wheel Torque	Nm	no

Table 10.5: Signals from CarMaker to GT-SUITE

Link ID	CarMaker Variable	GT-SUITE Signal Name	Unit	Required
58	Vehicle.RL2.Trq_T2W	Second Rear Left Wheel Torque	Nm	no
59	Vehicle.RR2.Trq_T2W	Second Rear Right Wheel Torque	Nm	no
60	Brake.Trq_tot[6]	Second Front Left Wheel Braking Torque	Nm	no
61	Brake.Trq_tot[7]	Second Front Right Wheel Braking Torque	Nm	no
62	Brake.Trq_tot[4]	Second Rear Left Wheel Braking Torque	Nm	no
63	Brake.Trq_tot[5]	Second Rear Right Wheel Braking Torque	Nm	no
64	Brake.IF.Trq_Reg_trg[6]	Second Front Left Wheel Target Regen Torque	Nm	no
65	Brake.IF.Trq_Reg_trg[7]	Second Front Right Wheel Target Regen Torque	Nm	no
66	Brake.IF.Trq_Reg_trg[4]	Second Rear Left Wheel Target Regen Torque	Nm	no
67	Brake.IF.Trq_Reg_trg[5]	Second Rear Right Wheel Target Regen Torque	Nm	no
68	Vehicle.FL2.Trq_WhlBearing	Second Front Left Wheel Bearing Torque	Nm	no
69	Vehicle.FR2.Trq_WhlBearing	Second Front Right Wheel Bearing Torque	Nm	no
70	Vehicle.RL2.Trq_WhlBearing	Second Rear Left Wheel Bearing Torque	Nm	no
71	Vehicle.RR2.Trq_WhlBearing	Second Rear Right Wheel Bearing Torque	Nm	no
101.. 512	Additional Signals (user defined)		no Unit	no

**Interface Imports: GT-SUITE -> CarMaker**

Table 10.6: Signals from GT-SUITE to CarMaker

Link ID	GT-SUITE Signal Name	CarMaker Variables	Unit	Required
1	Engine Speed	PowerTrain.IF.Engine_rotv	rad/s	yes
2	Current Gear No	PowerTrain.IF.GearNo	-	yes
3	Ignition State	PowerTrain.IF.Ignition	-	yes
4	Operation State	PowerTrain.IF:OperationState	-	yes
5	Front Left Wheel Speed	PowerTrain.IF.WheelOut[0].rotv	rad/s	yes
6	Front Left Wheel Speed	PowerTrain.IF.WheelOut[1].rotv	rad/s	yes
7	Front Left Wheel Speed	PowerTrain.IF.WheelOut[2].rotv	rad/s	yes
8	Front Left Wheel Speed	PowerTrain.IF.WheelOut[3].rotv	rad/s	yes
9	Front Left Wheel Position	PowerTrain.IF.WheelOut[0].rot	rad	no
10	Front Right Wheel Position	PowerTrain.IF.WheelOut[1].rot	rad	no
11	Rear Left Wheel Position	PowerTrain.IF.WheelOut[2].rot	rad	no
12	Rear Right Wheel Position	PowerTrain.IF.WheelOut[3].rot	rad	no
13	Front Left Wheel Reduced Brake Torque	PowerTrain.IF.WheelOut[0].Trq_B2WC	Nm	no
14	Front Right Wheel Reduced Brake Torque	PowerTrain.IF.WheelOut[1].Trq_B2WC	Nm	no
15	Rear Left Wheel Reduced Brake Torque	PowerTrain.IF.WheelOut[2].Trq_B2WC	Nm	no
16	Rear Right Wheel Reduced Brake Torque	PowerTrain.IF.WheelOut[3].Trq_B2WC	Nm	no
17	Front Left Wheel Drive Torque Support	PowerTrain.IF.WheelOut[0].Trq_Supp2WC	Nm	no
18	Front Right Wheel Drive Torque Support	PowerTrain.IF.WheelOut[1].Trq_Supp2WC	Nm	no
19	Rear Left Wheel Drive Torque Support	PowerTrain.IF.WheelOut[2].Trq_Supp2WC	Nm	no
20	Rear Right Wheel Drive Torque Support	PowerTrain.IF.WheelOut[3].Trq_Supp2WC	Nm	no
21	Front Left Wheel Drive Torque	PowerTrain.IF.WheelOut[0].Trq_Drive	Nm	no
22	Front Right Wheel Drive Torque	PowerTrain.IF.WheelOut[1].Trq_Drive	Nm	no
23	Rear Left Wheel Drive Torque	PowerTrain.IF.WheelOut[2].Trq_Drive	Nm	no
24	Rear Right Left Wheel Drive Torque	PowerTrain.IF.WheelOut[3].Trq_Drive	Nm	no

Table 10.6: Signals from GT-SUITE to CarMaker

Link ID	GT-SUITE Signal Name	CarMaker Variables	Unit	Required
25	Front Left Wheel Regen. Torque	PowerTrain.IF.WheelOut[0].Trq_BrakeReg	Nm	no
26	Front Right Wheel Regen. Torque	PowerTrain.IF.WheelOut[1].Trq_BrakeReg	Nm	no
27	Rear Left Wheel Regen. Torque	PowerTrain.IF.WheelOut[2].Trq_BrakeReg	Nm	no
28	Rear Right Wheel Regen. Torque	PowerTrain.IF.WheelOut[3].Trq_BrakeReg	Nm	no
29	Front Left Wheel Max. Regen. Torque	PowerTrain.IF.WheelOut[0].Trq_BrakeReg_max	Nm	no
30	Front Right Wheel Max. Regen. Torque	PowerTrain.IF.WheelOut[1].Trq_BrakeReg_max	Nm	no
31	Rear Left Wheel Max. Regen. Torque	PowerTrain.IF.WheelOut[2].Trq_BrakeReg_max	Nm	no
32	Rear Right Wheel Max. Regen. Torque	PowerTrain.IF.WheelOut[3].Trq_BrakeReg_max	Nm	no
33	Drive Torque Support on Body Fr1A (x)	PowerTrain.IF.Trq_Supp2Bdy1[0]	Nm	no
34	Drive Torque Support on Body Fr1A (y)	PowerTrain.IF.Trq_Supp2Bdy1[1]	Nm	no
35	Drive Torque Support on Body Fr1A (z)	PowerTrain.IF.Trq_Supp2Bdy1[2]	Nm	no
36	Drive Torque Support on Body Fr1B (x)	PowerTrain.IF.Trq_Supp2Bdy1B[0]	Nm	no
37	Drive Torque Support on Body Fr1B (y)	PowerTrain.IF.Trq_Supp2Bdy1B[1]	Nm	no
38	Drive Torque Support on Body Fr1B (z)	PowerTrain.IF.Trq_Supp2Bdy1B[2]	Nm	no
39	Drive Torque Support on Engine (x)	PowerTrain.IF.Trq_Supp2BdyEng[0]	Nm	no
40	Drive Torque Support on Engine (y)	PowerTrain.IF.Trq_Supp2BdyEng[1]	Nm	no
41	Final Drive Variable Gear Ratio	PowerTrain.IF.DL_iDiff_mean	-	no
42	Operation Error	PowerTrain.IF.OperationState	-	no
43	Strategy	PowerTrain.ControllIF.StrategyMode	-	no
55	Second Front Left Wheel Speed	PowerTrain.IF.WheelOut[0].rotv	rad/s	no
56	Second Front Right Wheel Speed	PowerTrain.IF.WheelOut[1].rotv	rad/s	no
57	Second Rear Left Wheel Speed	PowerTrain.IF.WheelOut[2].rotv	rad/s	no

Table 10.6: Signals from GT-SUITE to CarMaker

Link ID	GT-SUITE Signal Name	CarMaker Variables	Unit	Required
58	Second Rear Right Wheel Speed	PowerTrain.IF.WheelOut[3].rotv	rad/s	no
59	Second Front Left Position	PowerTrain.IF.WheelOut[0].rot	rad	no
60	Second Front Right Position	PowerTrain.IF.WheelOut[1].rot	rad	no
61	Second Rear Left Position	PowerTrain.IF.WheelOut[2].rot	rad	no
62	Second Rear Right Left Position	PowerTrain.IF.WheelOut[3].rot	rad	no
63	Second Front Left Wheel Reduced Brake Torque	PowerTrain.IF.WheelOut[6].Trq_B2WC	Nm	no
64	Second Front Right Wheel Reduced Brake Torque	PowerTrain.IF.WheelOut[7].Trq_B2WC	Nm	no
65	Second Rear Left Wheel Reduced Brake Torque	PowerTrain.IF.WheelOut[4].Trq_B2WC	Nm	no
66	Second Rear Right Wheel Reduced Brake Torque	PowerTrain.IF.WheelOut[5].Trq_B2WC	Nm	no
67	Second Front Left Wheel Drive Torque Support	PowerTrain.IF.WheelOut[6].Trq_Supp2WC	Nm	no
68	Second Front Right Wheel Drive Torque Support	PowerTrain.IF.WheelOut[7].Trq_Supp2WC	Nm	no
69	Second Rear Left Wheel Drive Torque Support	PowerTrain.IF.WheelOut[4].Trq_Supp2WC	Nm	no
70	Second Rear Right Wheel Drive Torque Support	PowerTrain.IF.WheelOut[5].Trq_Supp2WC	Nm	no
71	Second Front Left Wheel Drive Torque	PowerTrain.IF.WheelOut[6].Trq_Drive	Nm	no
72	Second Front Right Wheel Drive Torque	PowerTrain.IF.WheelOut[7].Trq_Drive	Nm	no
73	Second Rear Left Wheel Drive Torque	PowerTrain.IF.WheelOut[4].Trq_Drive	Nm	no
74	Second Rear Right Wheel Drive Torque	PowerTrain.IF.WheelOut[5].Trq_Drive	Nm	no
75	Second Front Left Wheel Regen. Torque	PowerTrain.IF.WheelOut[6].Trq_BrakeReg	Nm	no
76	Second Front Right Wheel Regen. Torque	PowerTrain.IF.WheelOut[7].Trq_BrakeReg	Nm	no
77	Second Rear Left Wheel Regen. Torque	PowerTrain.IF.WheelOut[4].Trq_BrakeReg	Nm	no

Table 10.6: Signals from GT-SUITE to CarMaker

Link ID	GT-SUITE Signal Name	CarMaker Variables	Unit	Required
78	Second Rear Right Wheel Regen. Torque	Power-Train.IF.WheelOut[5].Trq_BrakeReg	Nm	no
79	Second Front Left Wheel Max. Regen. Torque	Power-Train.IF.WheelOut[6].Trq_BrakeReg_max	Nm	no
80	Second Front Right Wheel Max. Regen. Torque	Power-Train.IF.WheelOut[7].Trq_BrakeReg_max	Nm	no
81	Second Rear Left Wheel Max. Regen. Torque	Power-Train.IF.WheelOut[4].Trq_BrakeReg_max	Nm	no
82	Second Rear Right Wheel Max. Regen. Torque	Power-Train.IF.WheelOut[5].Trq_BrakeReg_max	Nm	no
91	Inst. Fuel Consumption	PT.Control.Consump.Fuel.Act		no
92	Average Fuel Consumption	PT.Control.Consump.Fuel.Avg		no
93	Absolute Fuel Consumption	PT.Control.Consump.Fuel.Abs	I	no
94	Inst. Electrical Consumption	PT.Control.Consump.Elecl.Act		no
95	Average Electrical Consumption	PT.Control.Consump.Elecl.Avg		no
96	Absolute Electrical Consumption	PT.Control.Consump.Elec.Abs	kWh	no
97	State of Charge (Low Volt Battery)	PowerTrain.BatteryCU_IF.SOC_LV	%	no
98	State of Charge (High Volt Battery)	PowerTrain.BatteryCU_IF.SOC_HV	%	no
101..512	Additional Signals (user defined)		no Unit	no

## 10.5 Road: CarMaker - ADAS RP Interface

CarMaker supports the use of the ADAS Research Platform (ADAS RP) development environment by HERE. Modern energy management or novel driver assistance systems, in addition to the current driving mode and ambient information, also require predictions to be made about the anticipated characteristics of the route. They include, for example, uphill sections in the road, cornering radii, speed limits, number of lanes, intersections and traffic lights to optimize the controller strategy.

The support for ADAS RP contains the following features:

- Export from ADAS RP routes to IPGRoad including the three-dimensional description of the track and
  - multilane support
  - junctions
  - creation of an elevation profile from the map database elevation data
  - automatic creation of road markings
  - automatic creation of bridges
  - tunable smoothing of the reference line
  - automatic creation of the driving path along the calculated route
  - automatic detection of the driving side depending on the start of the route
  - export of traffic signs and traffic lights from the ADAS RP data base
  - UDAL link speed limits will be considered
- Communication interface using UDP protocol. This allows exchange of the vehicle's global position in CarMaker and the electronic horizon calculated by ADAS RP during the simulation.

Please note that the ADASRP plug-ins are available with a special license option only. The ADAS RP plug-ins are included in the CarMaker installation directory. No additional installation is required to run the interface, only a license extension.

### 10.5.1 Setting up the CarMaker - ADASRP Interface



Please note: Currently we support ADAS RP versions 2014 and 2015 for MS Visual Studio 10 (16.00). To check your installation go to the Help menu of ADAS RP and select *About ADAS RP*. The installation of MS Visual Studio is not required for the use of the CarMaker plug-ins.

Using ADAS RP with CarMaker 3.5.2 and above requires the following plug-ins for ADAS RP:

- CarMakerExport
- CarMakerRoad
- CarMakerSensorService
- CarMakerCommunication

Additionally an ADASRP.CM.ini file is provided which gives an example setup for ADAS RP.

#### Setup ADAS RP

Please find a list of supported ADAS RP version in the CarMaker Release Notes (mind the MS Visual Studio version!). In the CarMaker installation directory under `..\\IPG\\hil\\<arch-version>\\ADASRP-IF` choose the plugins matching with the installed ADAS RP version on your PC.

- ADASRP\_201x\_VSxx
- Copy the \*.dll and \*.ini files into the following directory:  
  <ADASRP-Inst.directory>/ADASRP/ADASRP.<version>/bin/Win32/ADASRP/
- Create a shortcut with the following settings for starting ADASRP:  
  Target:  
    <>/ADASRP/ADASRP.<version>/bin/Win32/  
    ADASRP/ADASRP.exe ADASRP.CM.ini  
  Execute in:  
    <>/ADASRP/ADASRP.<version>/bin/Win32/ADASRP

Depending on the ADAS RP version and the used map files it might be necessary to adjust the ADASRP.CM.ini file.

## Setup CarMaker for using the UDP Interface

Start the CarMaker GUI and go to *Application > Edit ‘SimParameters’* and add the following entries:

---

**ADASRP.Host = <hostname>**

---

Replace <hostname> by the name of your PC.

---

**ADASRP.Port = <number>**

---

This is the communication port mentioned in the CarMaker Communication Plug-in.

---

**ADASRP.ReceiveInterval = <time>**

---

This parameter defines the observation interval between received messages, default: 0.001s.

---

**ADASRP.SendInterval = <time>**

---

This is the interval between two sent messages for position and velocity, default: 0.1s.

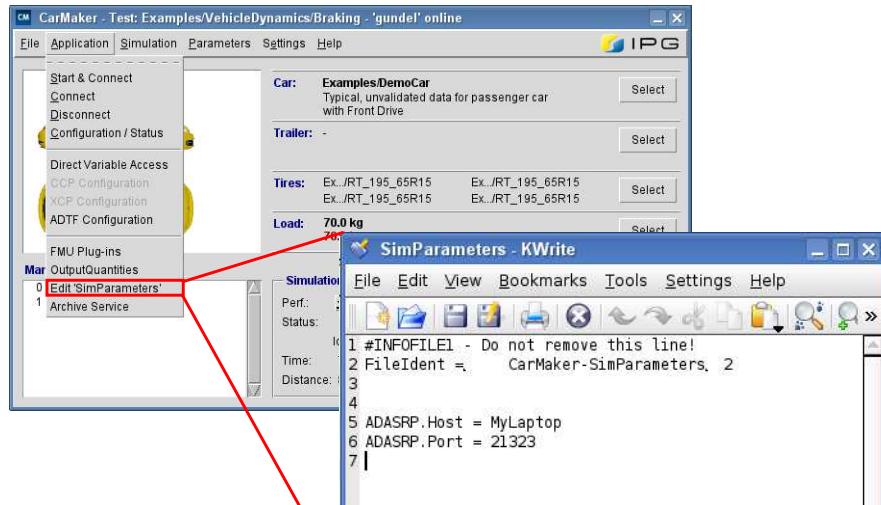


Figure 10.28: Modify the SimParameters

## 10.5.2 Export Routes from ADAS RP to CarMaker

### Export Simple Roads

To export a simple digitized road with one route or path only from ADAS RP to CarMaker proceed as follows:

- Choose a start and end position and calculate the route in ADAS RP
- Switch to the *CarMaker Route Export* panel.
- Eventually change the settings for the route export. [Table 10.7: CarMakerExport parameters](#) provides an overview of the available settings. For most application cases, you can keep the default values.
- Export the route by pressing the button *Create and Save CarMaker Road from Route*
- Select in the CarMaker road dialog the exported .road file as digitized road (see [section 4.4.4 'Digitized Roads' on page 47](#)).

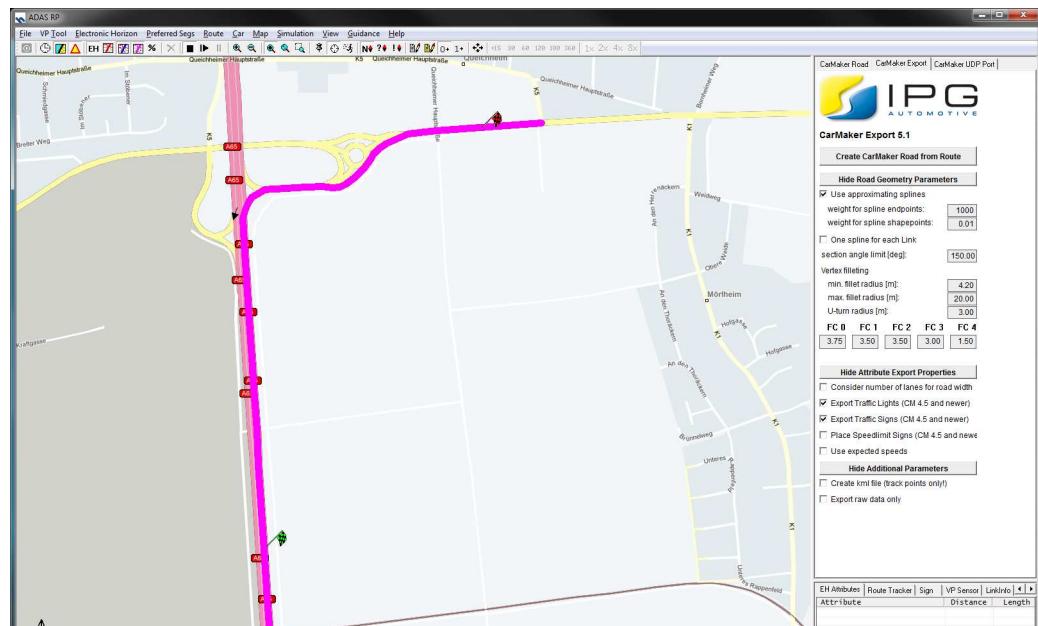


Figure 10.29: ADAS RP CarMakerExport

Table 10.7: CarMakerExport parameters

Parameter	Default	Description
Use approximating splines	checked	Defines if the calculated spline uses the calculated points for an approximation or if it goes directly through them.
Weight for spline end-points	1000	Sets the weight for start and endpoints of approx. spline
Weight for spline shape-points	0.01	Sets the weight of points of approx. spline between start and endpoint
One spline for each link	not checked	In general, the route is divided in multiple splines depending on the <i>section angle limit</i> parameter Check this option to create splines depending on the subdivision of roads into links by ADASRP, This can lead to less smooth road shapes.
Section angle limit [deg]	160	If the angle between three successive points is smaller than the limit defined here, a junction is placed at the middle point. This part of the route is then rounded appropriately to get a smooth turn.
Min./Max. fillet radius [m]	4.2/20	Minimum and maximum radius of the rounded turn. The actual radius depends on the section angle.
U-turn radius [m]	3	Radius of the semicircle that is used for representing a U-turn in the exported road
FC0... FC4	3.75 ... 2.5	Road width depending on the road class defined in ADASRP maps (freeway, highway, urban...)
Consider number of lanes for road width	not checked	Flag for considering the number of lanes for road width
Export Traffic Lights	checked	Flag for exporting traffic lights
Export Traffic Signs	checked	Flag for exporting traffic signs
Place Speedlimit Signs	not checked	Flag for placing speed limit signs where speed limits were recognized (ADAS RP does not contain the positions of "real" signs, the areas are determined internally by a logic)
Use expected speeds	not checked	Flag for using expected speeds as speed limits, if no speed limit areas are determined
Create kml file (track points only!)	not checked	Create a .kml file instead of .road file out of the digitized point list. A .kml file does not contain information about road width, traffic signs etc.
Export raw data only	not checked	Forces the export of the raw points out of ADAS RP. Can be used with the option <i>Create kml file</i>

## Export Road Networks

To export a complete road network for the IPGRoad 5.0 from ADAS RP, basically proceed as explained in the section [section 10.5.2 'Export Routes from ADAS RP to CarMaker'](#). The road is planned as a main route which is defined by a start and end point. Besides that, a *junction level* can be defined which determines the level of considered side-paths near the main path. E.g. junction level = 1 means, all roads that branch from the

selected route will be exported up to the next Junction that they end up in, and all roads that branch from this junction. Depending on the settings, a road network up to the number of junctions including their links (= parts of the road that connect two junctions) will be exported.

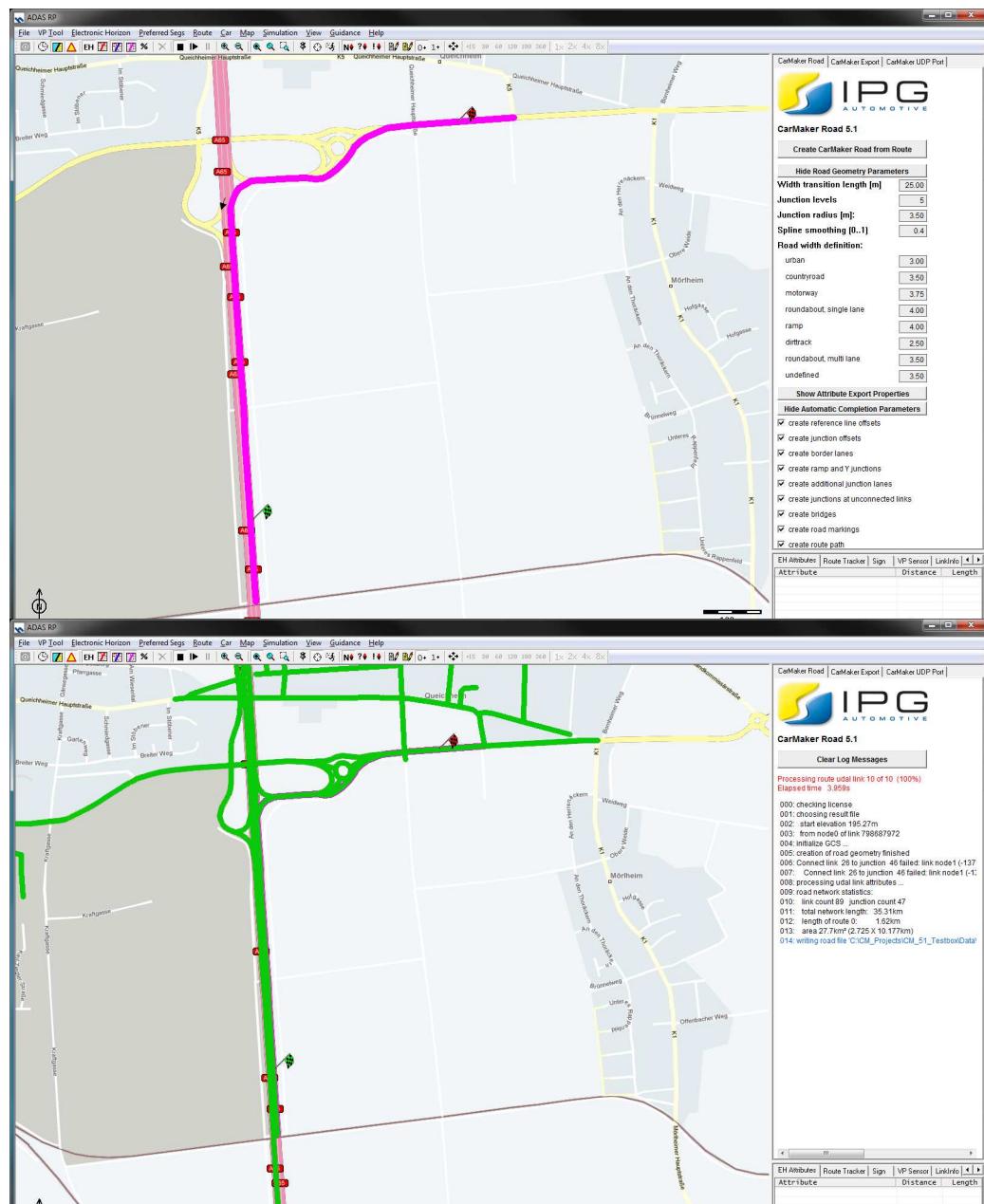


Figure 10.30: ADAS RP CarMakerRoad

The Table 10.8: [CarMakerRoad parameters](#) gives an overview of the parameters available.

Table 10.8: CarMakerRoad parameters

Parameter	Default	Description
Width transition length [m]	25	Length of road width transitions e.g. when changing from highway to an urban road
Junction levels	1	Describes the level of sidepaths considered on the main path of the road network

Table 10.8: CarMakerRoad parameters

Parameter	Default	Description
Junction radius [m]	3	Radius that is used to connect links to a junction
Spline smoothing (0..1)	0.2	Weight factor for the calculated spline
Road width definitions		Defines the road width depending on the road type (highway, urban, country road...)
Create reference line offsets	checked	Shift the driving lane so that the reference line is placed at the middle of all driving lanes
Create junction offsets	checked	Shift driving lanes of junctions in order to prevent a driving lane overlap
Create border lanes	checked	Create additional margins
Create ramp and Y junctions	checked	Check angle of junctions and define junctions type accordingly
Create additional junctions lanes	checked	Check and adjust the number of driving lanes depending on the junction type
Create junctions at unconnected links	checked	Check if multiple opened links cross each other and create junction accordingly
Create bridges	checked	Create bridges on crossing links
Create road markings	checked	Create road markings

### 10.5.3 Using the Online Data Exchange between ADAS RP and CarMaker

In case an geo-referenced road is used (e.g. an exported ADAS RP route), CarMaker can send the vehicle position via UDP to the CarMakerCommunication plug-in (see [Figure 10.32](#)). The CarMakerCommunication plug-in will pass the vehicle position to the CarMakerSensorService plug-in. This plug-in will then provide a GPS-signal for the ADAS-RP vehicle positioning (VP) module.

To enable this online feedback, select *COM Port* in the *ADAS RP VP Tool Sensor Service Configuration*.

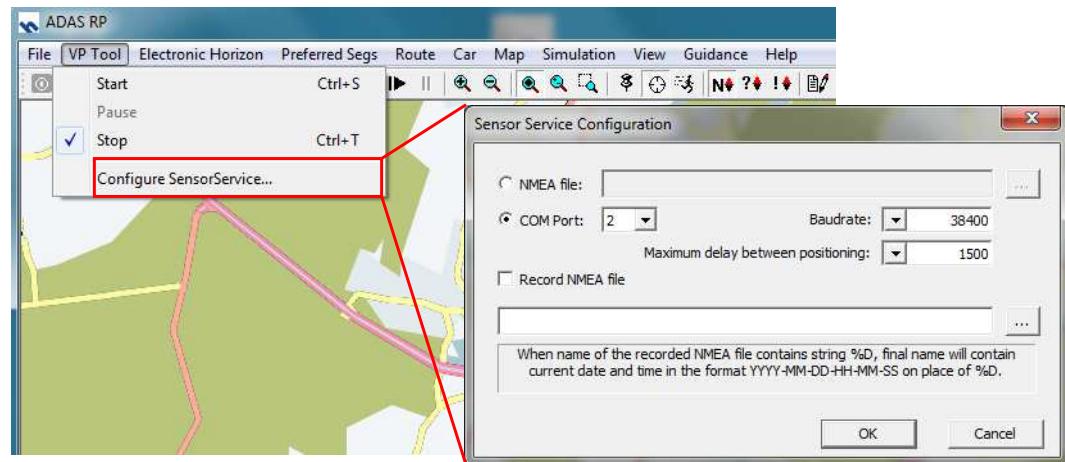


Figure 10.31: ADAS RP VP Tool Configuration

ADAS RP will then generate the electronic horizon. In case the electronic horizon is send via CAN the CarMakerCommunication plug-in will read the CAN messages and send them via UDP packages to CarMaker. On CarMaker side the received CAN message are buffered and can be read with the *ADASRP\_GetMsg()*-function (see template in *User.c*, *User\_In()*-function).

The data exchange is CAN protocol independent.

The *CarMaker UDP Port* module will automatically connect to the CarMaker application after the simulation is started and then start the vehicle positioning module. Once connected to a CarMaker session, other CarMaker sessions trying to connect are refused. After the simulation is stopped, the *CarMaker UDP Port* module will stop the vehicle positioning module and wait for any CarMaker application trying to connect.

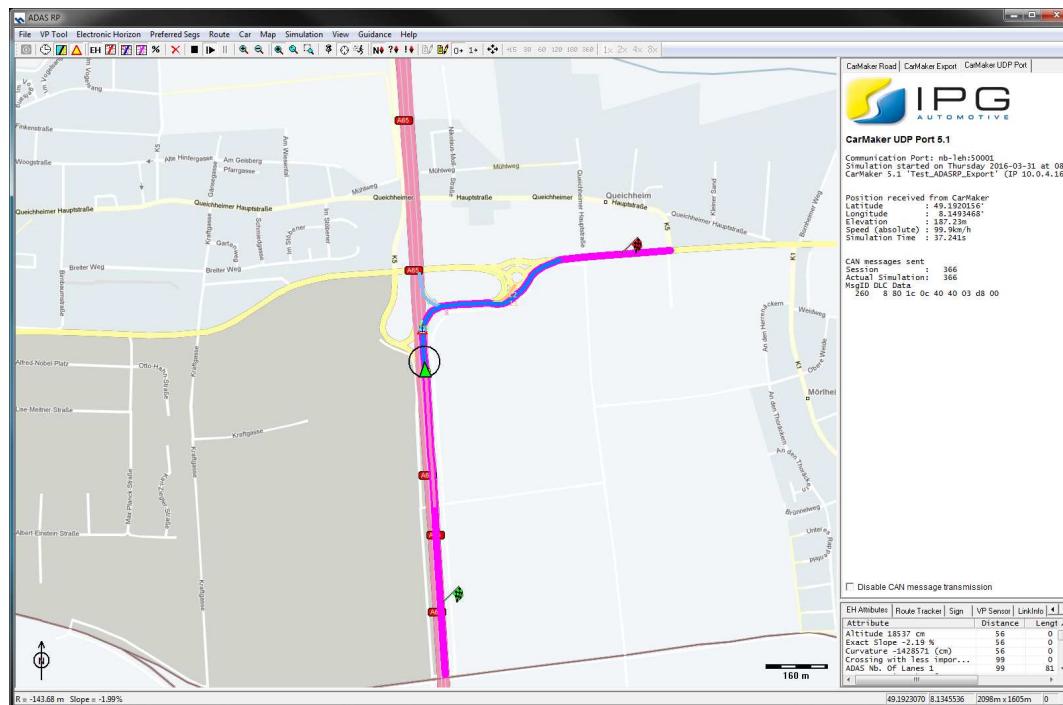


Figure 10.32: CarMaker - ADAS RP communication via UDP

## Appendix A

# FailSafeTester with CarMaker/HIL

Safety, reliability, efficiency, performance: These are the terms used by design engineers when discussing the implementation and design of automotive control units. However, as the knowledge and understanding in this field increases and new cutting edge ideas are built from the established concepts of yesterday, the complexity of the controller and the control algorithms that are used grows rapidly. As a result, it becomes very difficult or even impossible to predict the behavior of the controller in all situations, particularly when evaluated under real world conditions. Testing becomes necessary to determine whether or not the system's safety (and other) criteria can be met.

One important aspect of controller testing is the introduction of electrical faults into a system. This is necessary to simulate conditions that could occur with normal wear and tear, faulty installations, physical damage (e.g. from an accident), problematic sensors and actuators, or anything else that could potentially happen to a vehicle's electrical system over the course of its life.

The idea of testing for system failures is not new, but work in this area has traditionally been done by manually pulling wires and cables, and more recently with the help of breakout boxes that allow a bit more flexibility and incorporate some programmability and fault creation. But this sort of testing takes time, and time is money. IPG realized the need for a solution that is fast, easy and effective, and the result was the creation of the FailSafeTester.



Figure 10.33: The FailSafeTester

The idea is simple: create a product that can be used with little or no training, that is completely programmable, generates fully reproducible fault conditions, has unlimited possibilities for expansion, and fits in a space that is smaller than a lunch box. It should be able to create short circuits, cable breaks, current leaks, high current flows, and other conditions

that could arise due to changes in the controller's electrical environment, and also have an intuitive graphical user interface and be fully integrated into the CarMaker tool suite. IPG's FailSafeTester, as you might expect, does all of these things and more, and the rest of this section describes how it can be used.



If used improperly the FailSafeTester can cause damage to electronics in the ECU, I/O modules, etc. It should be used by someone who understands the effects that signal faulting will have.

Please refer to the FailSafeTester Reference Manual for detailed information on installation, hardware specifications, and other technical information not directly related to usage.

## A.1 How it Works

### A.1.1 Wiring Topology

#### Simple HIL Configuration

In a simple Hardware in the Loop (HIL) configuration using CarMaker/HIL, three hardware devices are connected with a series of cables or wires (see [Figure 10.34](#)).

- The first hardware device used in this simple setup is the host computer, which communicates with the real time computer over a network, and allows messages to be passed between the host computer and the real time computer. The tools such as the instruments panel, IPGMovie and IPGControl, etc. are able to send and receive information from the CarMaker executable running on the real time computer.
- The second piece of the puzzle is the real time computer that is running the CarMaker executable. The CarMaker executable has been configured in a way that allows communication through its I/O modules and network interface card.
- The third piece of hardware needed is the actual hardware in the loop element: one or more ECUs wired and mounted to a test bench. The test bench wiring allows the signals to be monitored and also connects the controller(s) to the I/O modules of the real time computer.

#### Host Computer

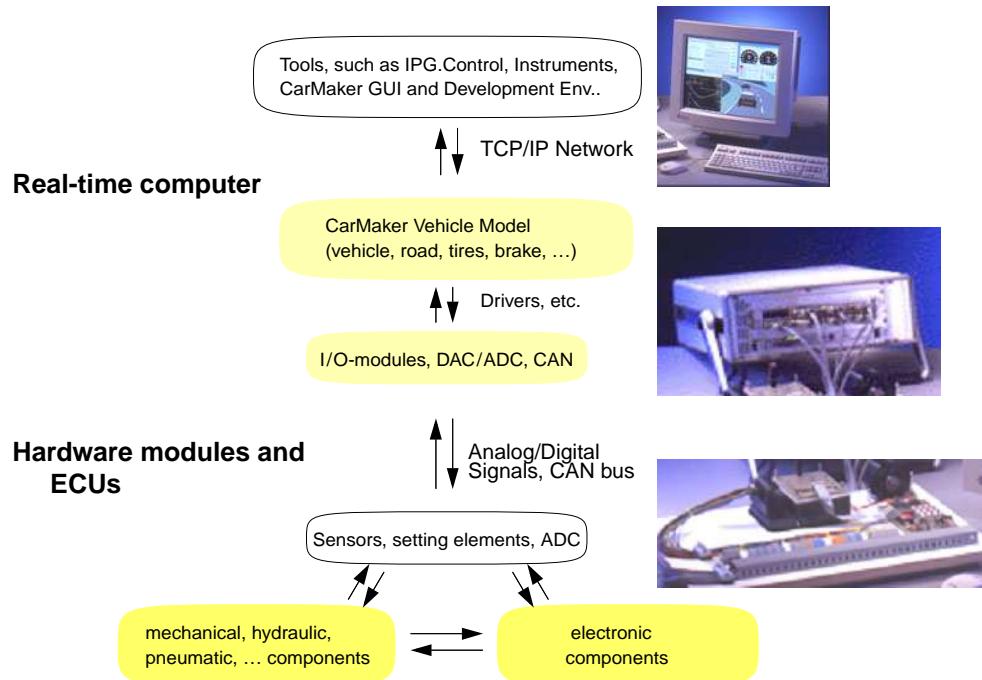


Figure 10.34: Standard HIL Setup

## HIL Configuration including FailSafeTester (1)

In the standard HIL configuration the hardware controller(s) is interfaced to the system sends and receives data exactly in the same style as if installed in a real vehicle. However, situations may occur in a real vehicle that effect the transmission of analog signals, power supplies, ground, etc. Wires could be damaged, corroded, shorted, crossed, cut, etc. and we want to be able to simulate these conditions. So, in order to do that we place a forth piece of hardware between the real time computer (acting as a virtual vehicle) and the test bench with the mounted and wired controller(s).

Through the fourth piece of hardware all signals chosen for testing can be run, and simulation of real world faults can be accomplished. The hardware used in this case is the FailSafeTester. [Figure 10.35](#) shows the configuration of a CarMaker/HIL simulation environment using the FailSafeTester.

- The first piece of hardware is the real time computer running the CarMaker executable. The executable is configured in way that not only allows communication between the host computer (network card) and test bench (I/O modules), but a third part is configured for communication with the FailSafeTester over a CAN bus.
- The second piece of hardware is the host computer. It is set up exactly as in the simple HIL configuration, except tools such as the FailSafeTester GUI or user defined scripts are used to control the actions of the FailSafeTester.
- The third piece of hardware is the test bench, which has been modified to allow a direct connection to the real time computer, or to the FailSafeTester, or to a combination of both the FailSafeTester and the real time system. The modification is done by adding or making changes to the cabling.
- The fourth piece of hardware is the FailSafeTester, which is configured to allow the selected signals to be passed from the test bench to the real time system. The signals can be unchanged and simply passed through unaffected, or modified in some way, as desired. The FailSafeTester is told what to do through CAN communication to the real time computer, which gets its instructions from user defined commands or from the host computer through the FailSafeTester dialog controlled by mouse clicks.

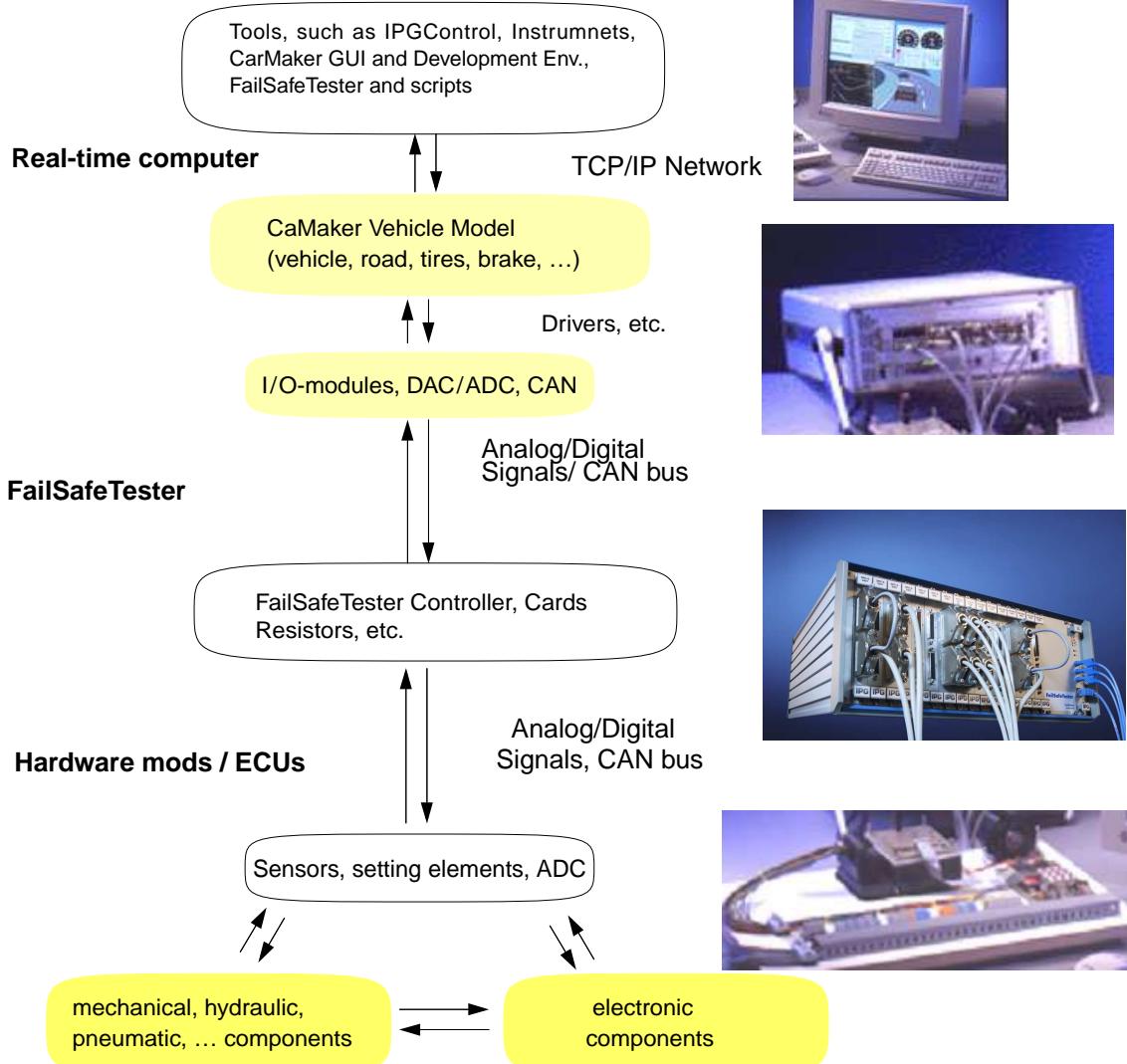
**Host Computer**

Figure 10.35: HIL setup with FailSafeTester

**HIL Configuration Including FailSafeTester (2)**

The FailSafeTester is not limited to the configuration explained in the previous section. There are a number of different ways it can be connected, depending on what hardware needs to be interfaced, or what signals should be run through the FailSafeTester hardware. For example, when simulating with multiple ECUs or numerous hardware components the FailSafeTester could be placed between the ECUs to modify signals that are exchanged between them.

## A.1.2 Inside the FailSafeTester

### Relays

The basic functionality is primarily provided by opening and closing relays. Opening a relay cuts a signal, closing it connects the signal. Like an on-off switch, the relays allow or disallow the flow of electrons from one point to the next. The relays used (SPST - Single Pole Single Throw), can have a starting position of either open or closed. [Figure 10.36](#) shows a signal line with a closed relay and the same signal that has been opened (cut).

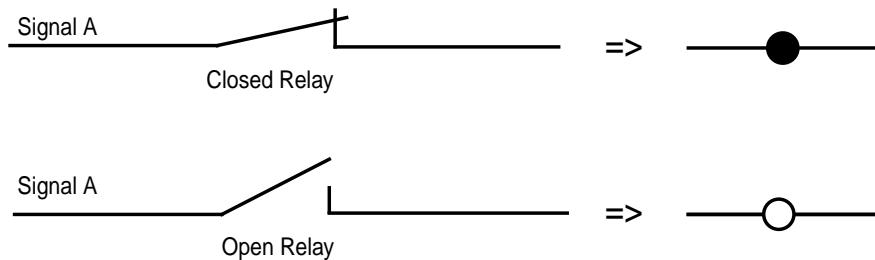


Figure 10.36: Signal with opened / closed relay

In the following figures a black filled circle stands for a closed relay whereas an unfilled circle indicates an opened relay.

### InterConnection Lines

Apart from cutting and reconnecting signals, the FailSafeTester can also cross signals, create short circuits, etc.. To explain this we need to introduce the idea of InterConnection lines. An InterConnection line can be thought of as a wire that runs perpendicular to the signal wires, and has a relay at the intersection point of the two wires. The relay is opened by default. [Figure 10.37](#) shows an example of this. Signal A and InterConnection Line 1 (IL1) are shown perpendicular to one another, with relay K1 at the intersection.

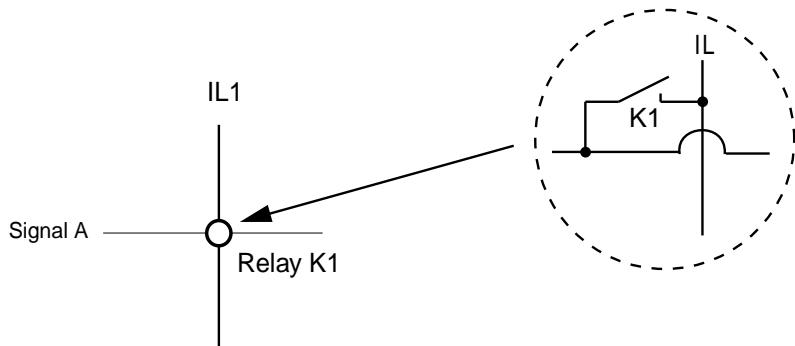
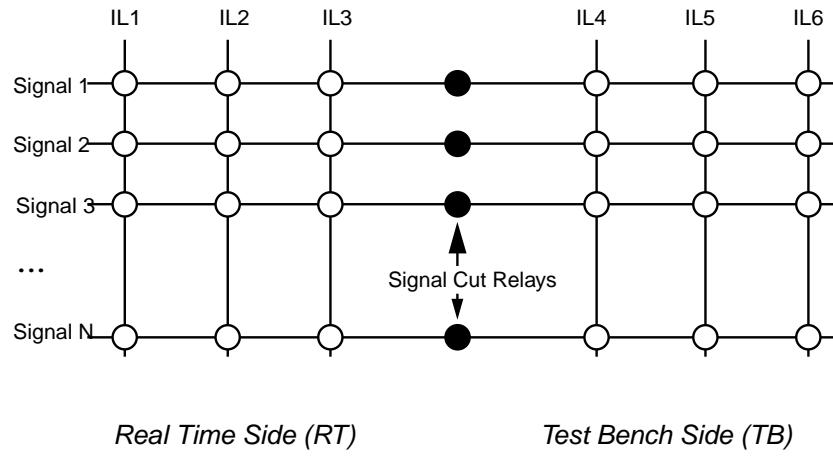


Figure 10.37: InterConnection Lines

[Figure 10.37](#) shows an example of an InterConnection Line - signal - relay grouping. The InterConnection line and signal are not connected by default, but if the relay is closed the two wires will be connected.

These InterConnection Lines are used to cross signals, create short circuits. For every signal that is passed through the FailSafeTester, there are a certain number of InterConnection Lines that can be used to connect them. Some of the InterConnection lines are placed on the test bench/ECU side of the FailSafeTester, and some on the real time computer side. [Figure 10.38](#) shows an example of this.



*Real Time Side (RT)*      *Test Bench Side (TB)*

Figure 10.38: InterConnection Lines / Signal Matrix

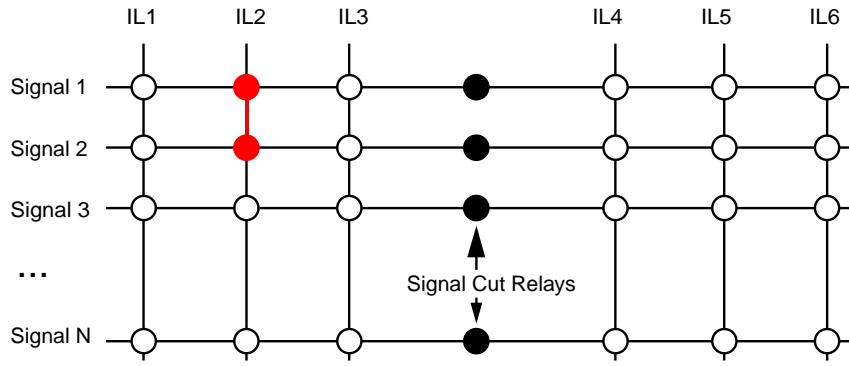


The real time side -, test bench side -, output side -, input side - terminology is an arbitrary naming convention that implies an enforced direction and can be a bit misleading. It might be easier to think of the FailSafeTester as having two sides, side A and side B. The direction is only relevant in relation to the hardware devices that are connected by the signals run through the FailSafeTester.

As [Figure 10.38](#) shows, signals and InterConnection lines can be connected in a number of ways. Let's see an example of this. The diagram in [Figure 10.39](#) shows the same matrix with slight modifications. Note the following changes:

- The *IL2\_RT - Signal 1* relay is closed.
- The *IL2\_RT - Signal 2* relay is closed.

The effect is a short circuit between signal 1 and signal 2. It is just as simple to switch signals and do any number of other things by opening and closing relays in the InterConnection Line / Signal Matrix. The red line shows the short-circuit line.



*Real Time Side (RT)*

*Test Bench Side (TB)*

Figure 10.39: Modified IL / Signal matrix

## Resistors

If one or more resistor cards are added to the FailSafeTester's hardware configuration then it is possible to add a resistance between signals or inside one signal via the InterConnection lines. The resistor can be used to simulate things like faulty cable insulation, corrosion, or other current leakage situations.

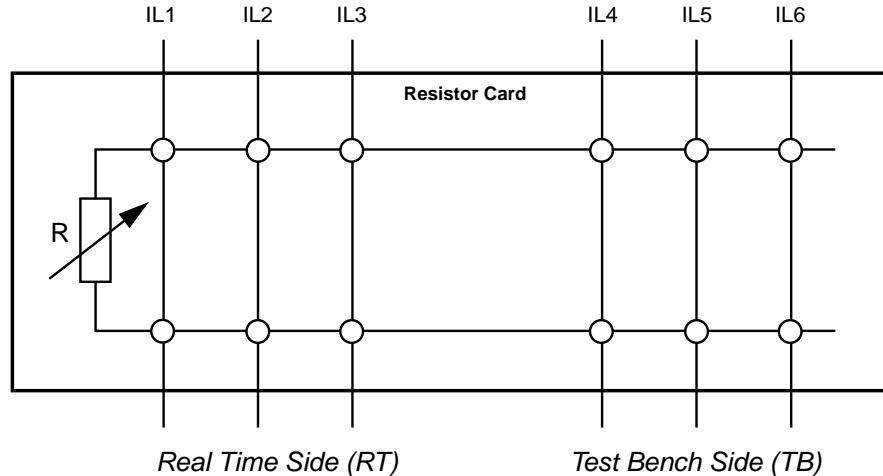


Figure 10.40: Inserting a Resistor

In Figure 10.41: the resistor of  $R=100\text{Ohms}$  is added to the signal to simulate an increased line resistance due to a corroded contact. The red line shows the current flow.

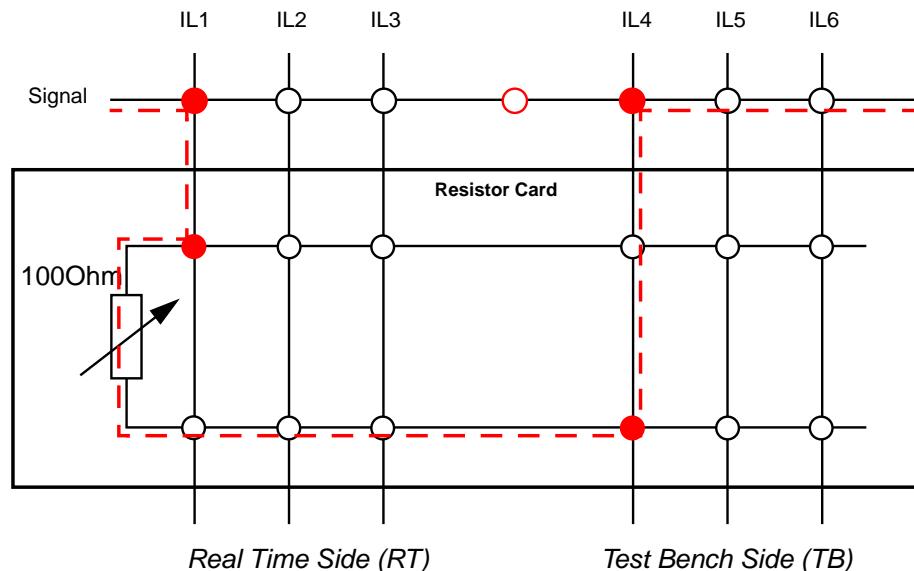


Figure 10.41: Example for Inserting an Resistor

## High Voltage Signals

The FailSafeTester's internal interconnection line don't support voltages of more than 60V. Therefore special high voltage cards are available which have their own interconnection line, the so-called *SUM Point* on the frontpanel. This *SUM Point* can be used to connect several High Voltage cards with each other. The following figure shows the connection matrix of those high voltage cards.

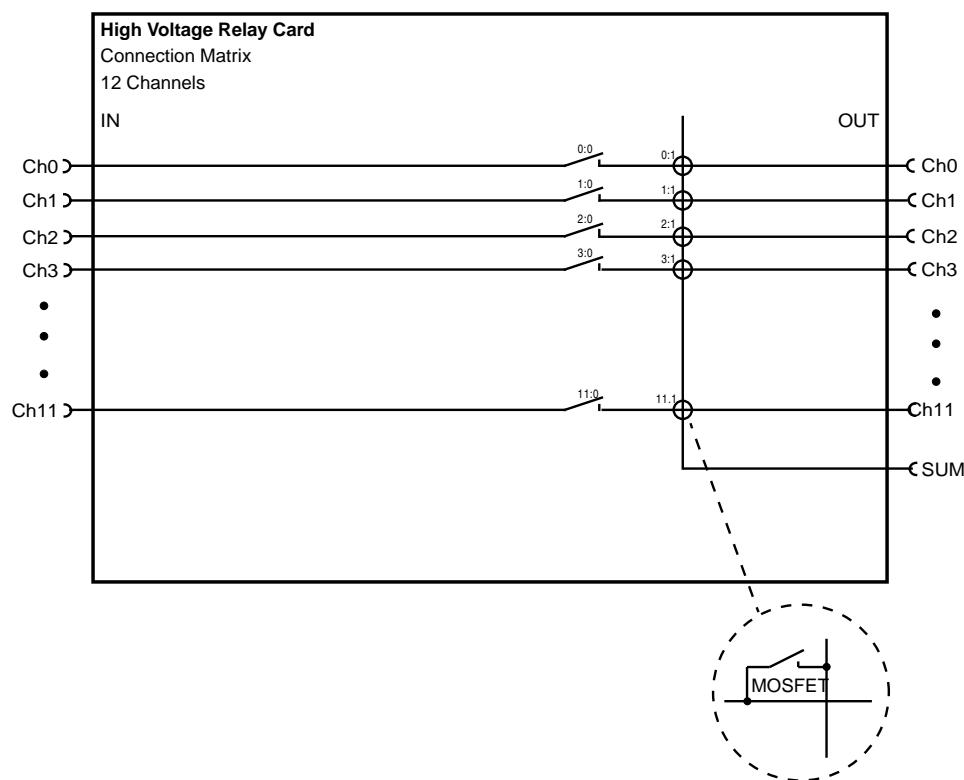
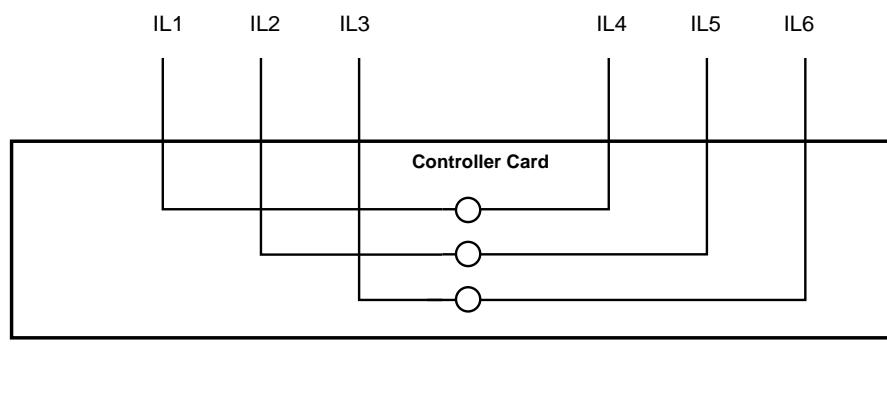


Figure 10.42: Connection Matrix of the High Voltage Card (HVC)

## Pairing of InterConnection Lines

Routing the input of one signal (Real Time side) to the output of another signal (Test Bench side) without generating a short-circuit between the 2 signals is possible by pairing a RT-InterConnection line with its corresponding TB-InterConnection line.



Real Time Side (RT)

Test Bench Side (TB)

Figure 10.43: Pairing Corresponding ILs

Figure 10.44: shows an example for crossing two signals. Therefore, the pairing of 4 ILs is needed. The red and blue lines show the current flow of the crossed signals.

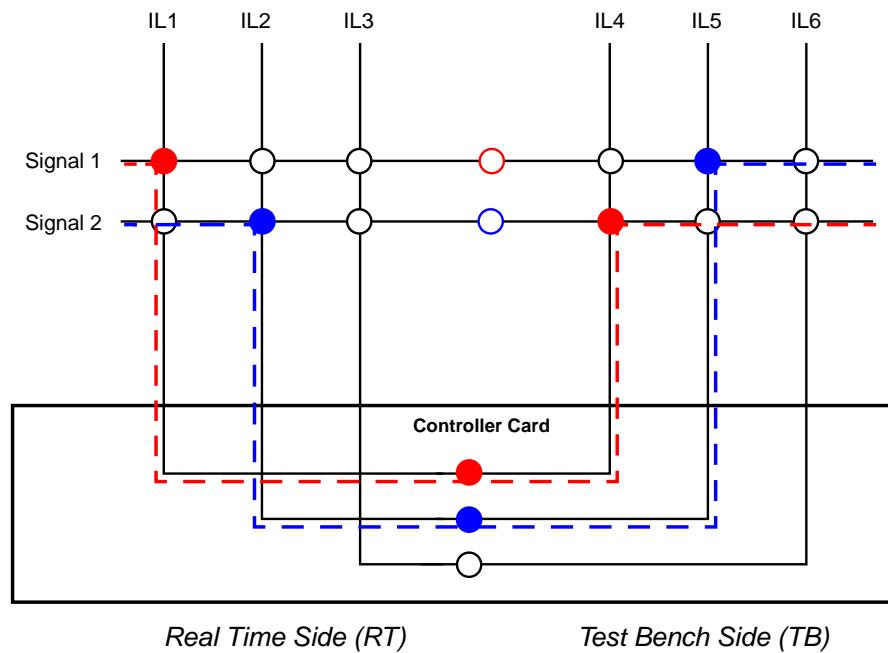


Figure 10.44: Crossing two signals

## A.2 FailSafeTester Cards

There are various cards available that can be used with the FailSafeTester, each providing a different functionality. Please find below a brief description, for further details please refer to the FailSafeTester Reference Manual.

### Controller Card (FST-CC)

The controller card communicates with the host PC and configures the switches inside the FailSafeTester. The controller card communicates via CAN or via RS232.

Since revision 2.0 also the self diagnosis functionality is implemented on the controller card. It checks for defective relays, micro fuses and other faults. Since the Controller

### Standard Relay Cards (FST-SRC)

The standard relay cards (SRCs) contain the relays that are needed to create the IC / Signal matrix explained in [section A.1.2](#). They are suitable for handling normal signal currents.

### High Current Relay Cards (30Amps, FST-HCRC)

High current relay cards (HCRCs) perform the same action as the SRCs, but are built to withstand high loads. Because of the size of the components, only four of the six standard InterConnection lines are available with HCRCs. Please refer to the FailSafeTester manual for more detailed information.

### High Current Relay Cards (50Amps, FST-RC50A)

High current relay cards (RC50A) perform the same action as the SRCs, but are built to withstand high loads. As with the HCRC because of the size of the components, only four of the six InterConnection lines are available with the RC50A. Please refer to the FailSafeTester manual for more detailed information

### Programmable Resistor Card (FST-PRC)

Resistor cards, as the name implies, contain resistors that can be connected to signals using the FailSafeTester's InterConnection Lines. Refer to the FailSafeTester manual for details.

### High Voltage Card (FST-HVC)

The FST-HVC allows switching of AC/DC voltages up to 150V and up to 15A. In opposite to other FST cards the FST-HVC has no connection to the internal interconnection lines of the FailSafeTester but has a special connection matrix which includes an external so-called SUM point which is available on the front panel. Refer to the FailSafeTester manual for details.

### Diagnosis Card (DIAG)

Due to the fact that the self diagnosis functionality is implemented on the controller card (FST-CC) since Revision 2.0 the Diagnosis Card (DIAG) is no longer available anymore. However older FST-DIAG card will still be supported in future.

## A.3 FailSafeTester Configuration GUI

In [section A.2 'FailSafeTester Cards' on page 568](#) we discussed the types of cards that can be used with the FailSafeTester, and in this section we describe how to configure those cards in the CarMaker environment.

It is recommended to configure the FailSafeTester via the FailSafeTester configuration dialog. In this window the CAN communication and the signal names are defined. The dialog is available in the CarMaker GUI under Realtime System > FailSafeTester.

As long as the FailSafeTester is not configured correctly or the real time executable is not connected to the CarMaker GUI, the FailSafeTester dialog looks as follows:



Figure 10.45: The FailSafeTester dialog if no FST is connected

Click on "Configure" to open the FailSafeTester configuration dialog.

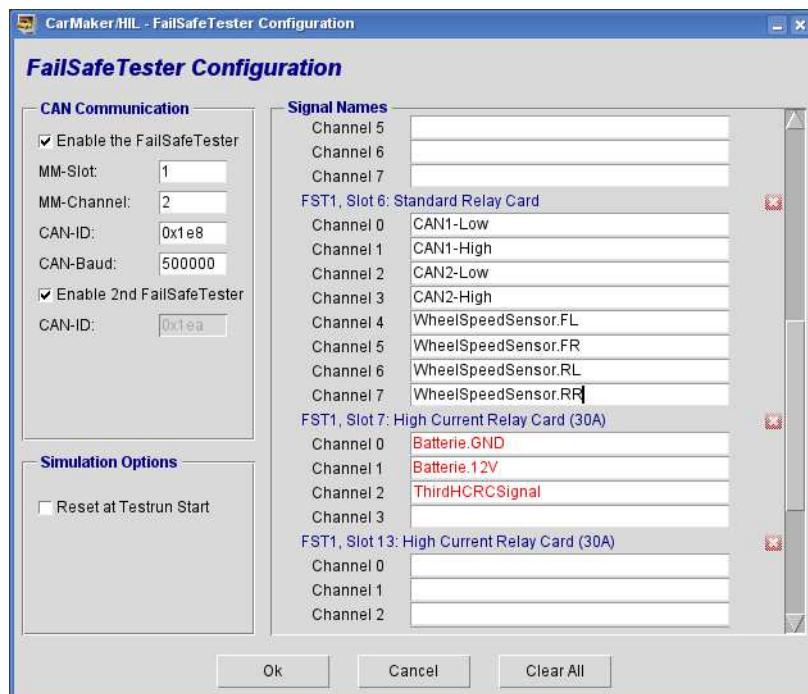


Figure 10.46: The FailSafeTester's Configuration dialog

As long as the CAN connection between FailSafeTester and the CarMaker has not been established, only the communication part of the FailSafeTester configuration dialog is available. Here you can configure the CAN slot, CAN channel and CAN-ID.

Once the connection is established the dialog shows a list of all SRC, HCRC and RC50A cards that are available within the FailSafeTester. Here you can configure an arbitrary signal name for each channel.

- Cancel** Closes the dialog without saving any changes.
- Ok** Saves the communication parameters and the signal names configured in the project's *ECUParameters* file.

- Clear All** Deletes all signal names.

If no signal names are configured, e.g. after first pressing [Clear All] and then [Save], all channels will be given a name with the following format:

<CardType><Slot>. <Channel>

If default names are used, they do not appear in the configuration dialog.

If a second FailSafeTester is connected and should be used, the second one must be configured to the CAN-ID of the first FST incremented by 2.

The signal names of the second FST can be found below the signal names of the first FST.



Mind: Once the communication parameters have changed, the real time executable must be restarted!



Note: The *ECUParameters* file is read by CarMaker only when starting or stopping a simulation. When changing the settings be sure to start or stop a TestRun to allow the changes to take effect.

If configured cards are removed from the FailSafeTester the entry fields of their signal names are disabled. However, the signal names of this card are stored until the card is deleted manually by clicking the red cross on the right border of the card description (see [Figure 10.47](#)).

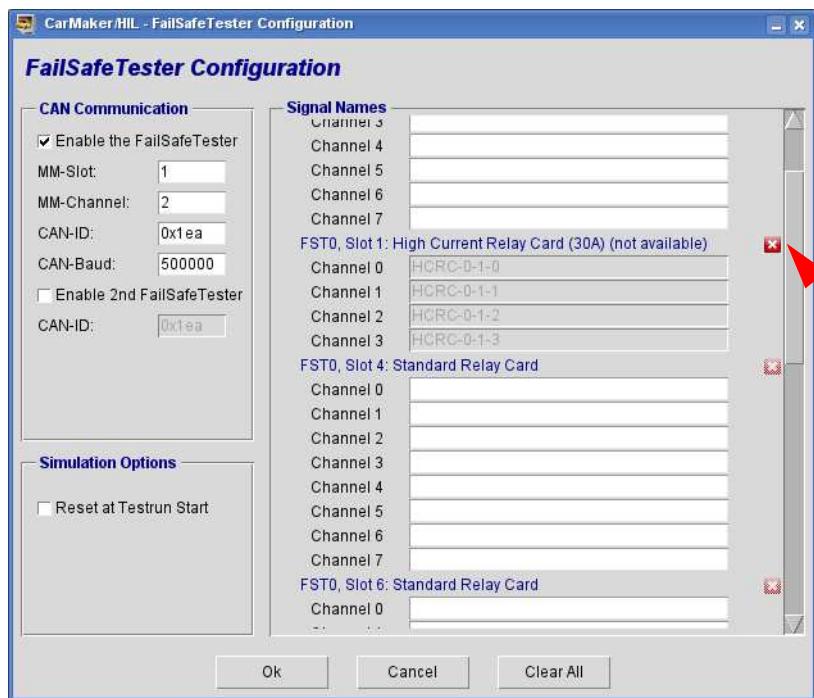


Figure 10.47: Deleting the Configuration of Removed Cards

The FST-HVC can also be configured in this GUI.

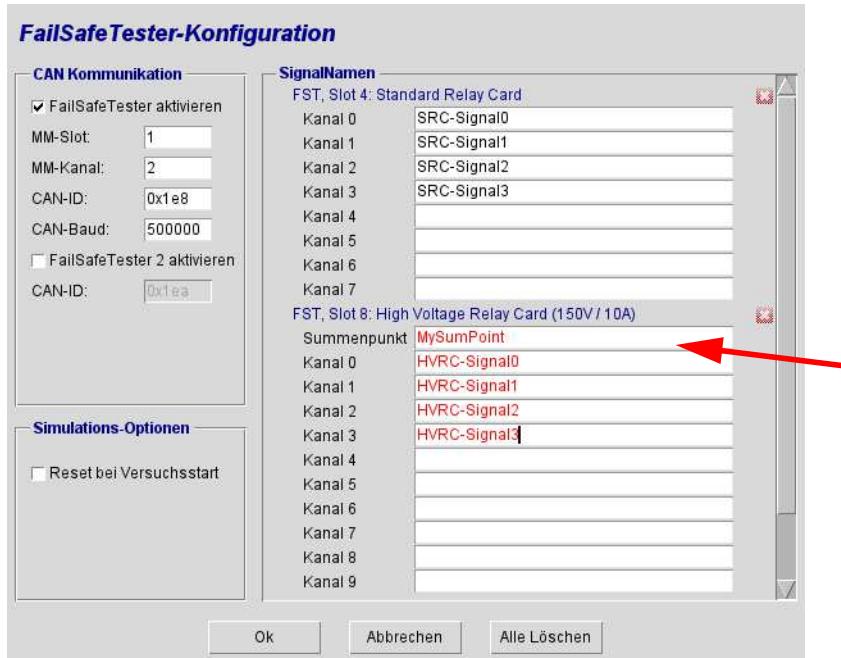


Figure 10.48: Configuring the HVC Card

The *SUM point* always must have a name. Otherwise the signals on this card cannot be used inside the *FailsafeTester* dialog.

It is possible to give the same name to the *SUM* points of more than one *FST-HVC*. This is useful if the *SUM* points of several *FST-HVC* are connected to each other. In this case the signals of all cards with the same *SUM* point are grouped and available in the same *FST-HVC* frame within the dialog. The number of *SUM* points with **different** names determine how many *FST-HVC* frames can be shown.

The *FST-HVC* frames only appear if there are *FST-HVC* cards plugged into the FST. Likewise default signal rows only appear, if *FST-SRCs* or *FST-HCRCs* are plugged into the *FailSafeTester*.

## A.4 Manual FailSafeTester Configuration

Of course it is still possible to configure the FailSafeTester directly editing the *ECUParameters* file that is located in the CarMaker project directory under *Data/Config/<Real Time Computer's Name>/ECUParameters*.



The *ECUParameters* file is read by CarMaker only when starting or stopping a simulation. When changing the settings be sure to start or stop a TestRun to allow the changes to take effect.

### A.4.1 Card Naming Convention

The following table shows the card IDs that must be configured to use the FailSafeTester channels. The IDs are used to specify the card type in the *ECUParameters* file. Refer to the FailSafeTester manual for more details concerning the types (and subtypes) of cards available.

Table 10.9: FailSafeTester Card types

CardType	Description
0x04	Standard Relays Card, 5Amps, 8 channels
0x05	High Current Relays Card, 30Amps, 4 channels
0x06	High Current Relays Card, 50Amps, 4 channels

Self diagnosis cards (DIAG), programmable resistor cards (PRC) and other card types that are not listed above must not be configured but are used if they are plugged into the FailSafeTester. If configured their parameters will be ignored.

### A.4.2 Configuration Settings

The following settings have to be added to the *ECUParameters* file. They are used to specify the required FailSafeTester configuration.

#### Required Configuration Settings

---

##### FST.CANSlot FST1.CANSlot

---

The slot of the CAN module to which the FailSafeTester is connected.

**Example**      `FST.CANSlot = 2`

If this value is set to -1 or is not available the FailSafeTester is not used.

`FST.xxx` is the configuration for the first FST.

`FST1.xxx` is the configuration for the second FST (if available).

---

**FST.CANChannel**  
**FST1.CANChannel**

---

The channel of the CAN module to which the FailSafeTester is connected.

**Example**    FST.CANSslot = 2

If this value is set to -1 or is not available the FailSafeTester is not used.

FST.xxx is the configuration for the first FST.

FST1.xxx is the configuration for the second FST (if available).

---

---

**FST.CANID**  
**FST1.CANID**

---

CAN ID of the FST controller card. The FST uses two subsequent CAN IDs. The first one is the ID for the command, the second one is used for answers from the FST to CarMaker. Only the first CAN ID (command ID) must be configured.

**Example**    FST.CAN.Id = 0x1e8

FST.xxx is the configuration for the first FST.

FST1.xxx is the configuration for the second FST (if available).

---

---

**FST.Cards**  
**FST1.Cards**

---

Number of cards to be used.

**Example**    FST.Cards = 5

If this value is set to 0 or is not available default names with the following format are used:  
<CardType><Slot>. <Channel>

FST.xxx is the configuration for the first FST.

FST1.xxx is the configuration for the second FST (if available).

---

---

**FST.Crd<CardNo>.Slot**  
**FST1.Crd<CardNo>.Slot**

---

Specifies the slot number, the card is plugged in.

**Example**    FST.Crd0.Slot = 2

FST.xxx is the configuration for the first FST.

FST1.xxx is the configuration for the second FST (if available).

---

**FST.Crd<CardNo>.Type**  
**FST1.Crd<CardNo>.Type**

---

For card types, see [section A.4.1 'Card Naming Convention'](#).

**Example**    FST.Crd0.Type = 0x04

FST.xxx is the configuration for the first FST.

FST1.xxx is the configuration for the second FST (if available).

---

**FST.Crd<CardNo>.Ch<No>**  
**FST1.Crd<CardNo>.Ch<No>**

---

Signal name of the specified channel. If this parameter is not given and [FST.Cards](#) [FST1.Cards](#) > 0 the channel can not be used.

---

**FST.SkipNewIfCfgUnchanged**  
**FST1.SkipNewIfCfgUnchanged**

---

Set to "1" the FailSafeTester(s) is/are will not be reset at the start of a simulation if signal names have not changed. By default this value is set to "0" which means, the FST is reset when a simulation starts.

**Example**    FST.SkipNewIfCfgUnchanged = 1

FST.xxx is the configuration for the first FST.

FST1.xxx is the configuration for the second FST (if available).

**Example**    ECUParameters configuration example for one FST

## Listing 10.1: ECUParameters example

```
1: #INFOFILE1.1 - Do not remove this line!
2: FileIdent =CarMaker-ECUParameters3
3:
4:
5: ### Activate the FailSafeTester with 2 relay cards
6: ### The PRC (0x07) in Slot 13 must not be configured
7: FST.CANSlot = 2
8: FST.CANChannel = 2
9: FST.CANID = 0x1e8
10:
11: FST.Cards = 3
12:
13: ### First Card: Relais (0x04)
14: ### in slot 0, with 8 channels; channel 2 is unused
15: FST.Crd0.Slot = 0
16: FST.Crd0.Type = 0x04
17: FST.Crd0.Ch0 = WhlSpd.FL
18: FST.Crd0.Ch1 = WhlSpd.FR
19: FST.Crd0.Ch3 = UBat
20: FST.Crd0.Ch4 = CAN.Low
21: FST.Crd0.Ch5 = WhlSpd.RL
22: FST.Crd0.Ch6 = WhlSpd.RR
23: FST.Crd0.Ch7 = CAN.High
24:
25:
26: ### Second Card: RelaisHighCurrent (0x05)
27: ### in slot 11, 4 channels,
28: FST.Crd1.Slot = 11
29: FST.Crd1.Type = 0x05
30: FST.Crd1.Ch0 = RlsHC.0
31: FST.Crd1.Ch1 = RlsHC.1
32: FST.Crd1.Ch2 = RlsHC.2
33: FST.Crd1.Ch3 = RlsHC.3
34:
```

## ECUParameters configuration example for two FSTs

Listing 10.2: ECUParameters Example

```
1: #INFOFILE1.1 - Do not remove this line!
2: FileIdent =CarMaker-ECUParameters3
3:
4: FST.CANID = 0x1ea
5: FST.CANSlot = 1
6: FST.CANChannel = 2
7: FST.SkipNewIfCfgUnchanged = 1
8: FST.Cards = 3
9: FST.Crd0.Ch0 = HCRC-0-1-0
10: FST.Crd0.Ch1 = HCRC-0-1-1
11: FST.Crd0.Ch2 = HCRC-0-1-2
12: FST.Crd0.Ch3 = HCRC-0-1-3
13: FST.Crd0.Slot = 1
14: FST.Crd0.Type = 0x05
15: FST.Crd1.Ch0 = CAN1.Low
16: FST.Crd1.Ch1 = CAN2.Low
17: FST.Crd1.Ch2 = CAN1.High
18: FST.Crd1.Ch3 = CAN2.High
19: FST.Crd1.Slot = 6
20: FST.Crd1.Type = 0x04
21: FST.Crd2.Ch0 = Ubat
22: FST.Crd2.Ch1 = GND
23: FST.Crd2.Ch2 = KL.15
24: FST.Crd2.Slot = 7
25: FST.Crd2.Type = 0x05
26: FST1.CANID = 0x1ea
27: FST1.CANSlot = -1
28: FST1.CANChannel = -1
29: FST1.SkipNewIfCfgUnchanged = 1
30: FST1.Cards = 3
31: FST1.Crd0.Ch0 = SRC-1-0-0
32: FST1.Crd0.Ch1 = SRC-1-0-1
33: FST1.Crd0.Ch2 = SRC-1-0-2
34: FST1.Crd0.Ch3 = SRC-1-0-3
35: FST1.Crd0.Ch4 = SRC-1-0-4
36: FST1.Crd0.Ch5 = MyBestChannel
37: FST1.Crd0.Slot = 0
38: FST1.Crd0.Type = 0x04
39: FST1.Crd2.Ch0 = GND
40: FST1.Crd2.Ch1 = Ubat
41: FST1.Crd2.Ch2 = KL.15
42: FST1.Crd2.Slot = 7
43: FST1.Crd2.Type = 0x05
44: FST1.Crd1.Ch0 = CAN1-Low
45: FST1.Crd1.Ch1 = CAN1-High
46: FST1.Crd1.Ch2 = CAN2-Low
47: FST1.Crd1.Ch3 = CAN2-High
48: FST1.Crd1.Slot = 6
49: FST1.Crd1.Type = 0x04
```

## A.5 FailsafeTester C-Interface

It is necessary to include function calls inside the CarMaker executable to allow the FailsafeTester to be initialized, and also to facilitate communication (via CAN bus) between the FailsafeTester and the real time computer.

### A.5.1 Global Variables

The following table shows the global (integer) variables that need to be assigned for the CarMaker application:

FST_CAN_Slot	slot of the M51 module
FST_CAN_Ch	CAN channel of the module
FST_CAN_Id	CAN ID of the FailsafeTester. FST_CAN_Id and FST_CAN_Id+1 are used for communication.

#### Example

Listing 10.3: FST Global Variables

```
1: FST_CAN_Slot = 2;
2: FST_CAN_Id = 0x1e8;
3: FST_CAN_Ch = 0;
4: FST_CAN_SendExt = MIO_M51_SendExt;
5: FST_CAN_RecvExt = MIO_M51_RecvExt;
6: FST_CAN_SendStat = MIO_M51_SendStat;
```

### A.5.2 Configure the M51 CAN module

Configure the M51 I/O module for 500kBaud and enable both CAN IDs as follows:

```
MIO_M51_SetCommParam (FST_CAN_Slot, FST_CAN_Ch, 500000, 70, 2, 0);
MIO_M51_EnableIds (FST_CAN_Slot, FST_CAN_Ch, FST_CAN_Id, 2);
```

## A.6 Usage

Up to this point we have talked about how to configure the FailSafeTester in the CarMaker environment ([section A.3 'FailSafeTester Configuration GUI' on page 569](#)) and briefly reviewed the currently available card types ([section A.2 'FailSafeTester Cards' on page 568](#)). We have also explained the wiring topology and the concept of switching relays and connecting signals to InterConnection lines ([section A.1 'How it Works' on page 560](#)), which are, when taken together, the fundamental concepts needed to understand how connections, short circuits, current leaks, etc. are realized with the FailSafeTester hardware. We now turn our discussion to practical applications. In other words, how to use the FailSafeTester in the CarMaker environment.

There are currently two different approaches to use the FailSafeTester in the CarMaker environment. First, by using the FailSafeTester dialog that is completely integrated into the CarMaker tool suite. The second method is to send script commands in the mini-maneuver command language. We will discuss both of these approaches in turn.

### A.6.1 FailSafeTester Dialog

#### Overview

To start using the FailSafeTester please make sure that all steps described in the former chapters were followed:

- the FailSafeTester is connected between a test bench with one or more ECUs and the real time computer that is running the CarMaker vehicle model
- the FailSafeTester card settings are specified in the *ECUParameters* file
- the FailSafeTester C functions are included in the CarMaker executable

The next step is to start using the FailSafeTester dialog by opening it from the real time system drop down menu in the CarMaker GUI. [Figure 10.49](#) shows the menu entry. Select it to open the FailSafeTester dialog.

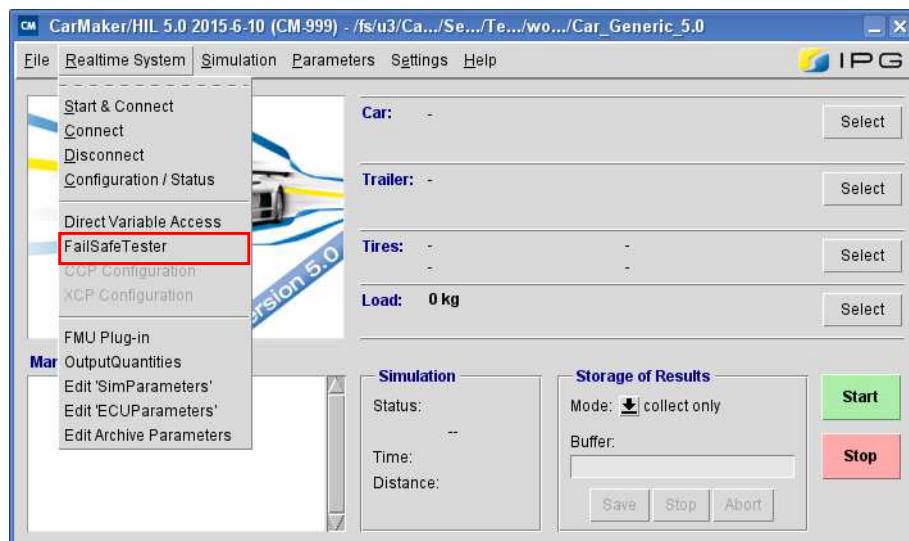


Figure 10.49: FailSafeTester in the CarMaker GUI

The windows which pops up looks as follows, in case one FailSafeTester is used:

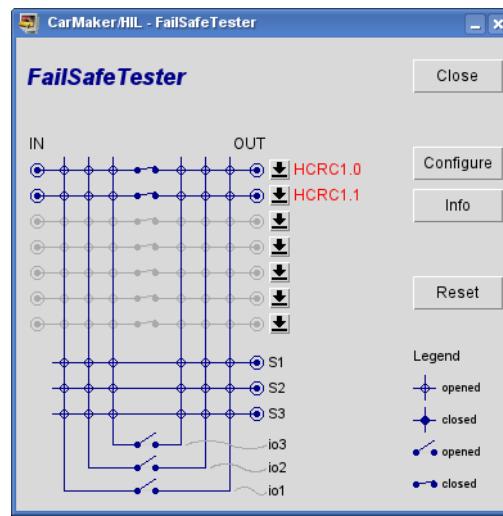


Figure 10.50: The FailSafeTester Dialog for 1 FST

If two FailSafeTesters are used the dialog looks as follows:

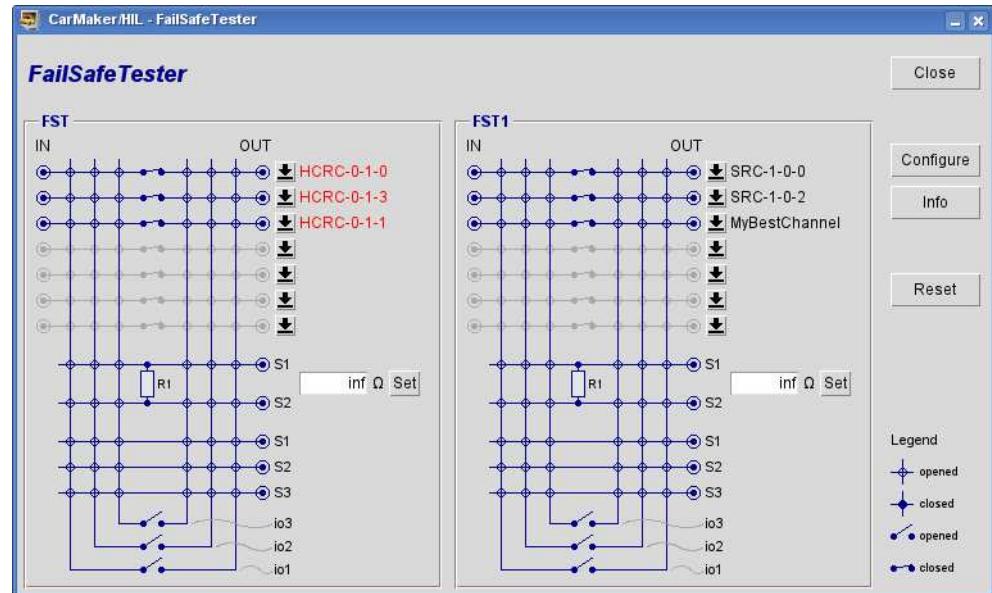
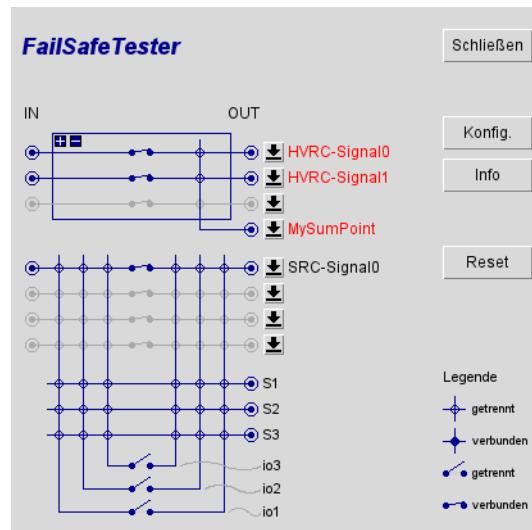


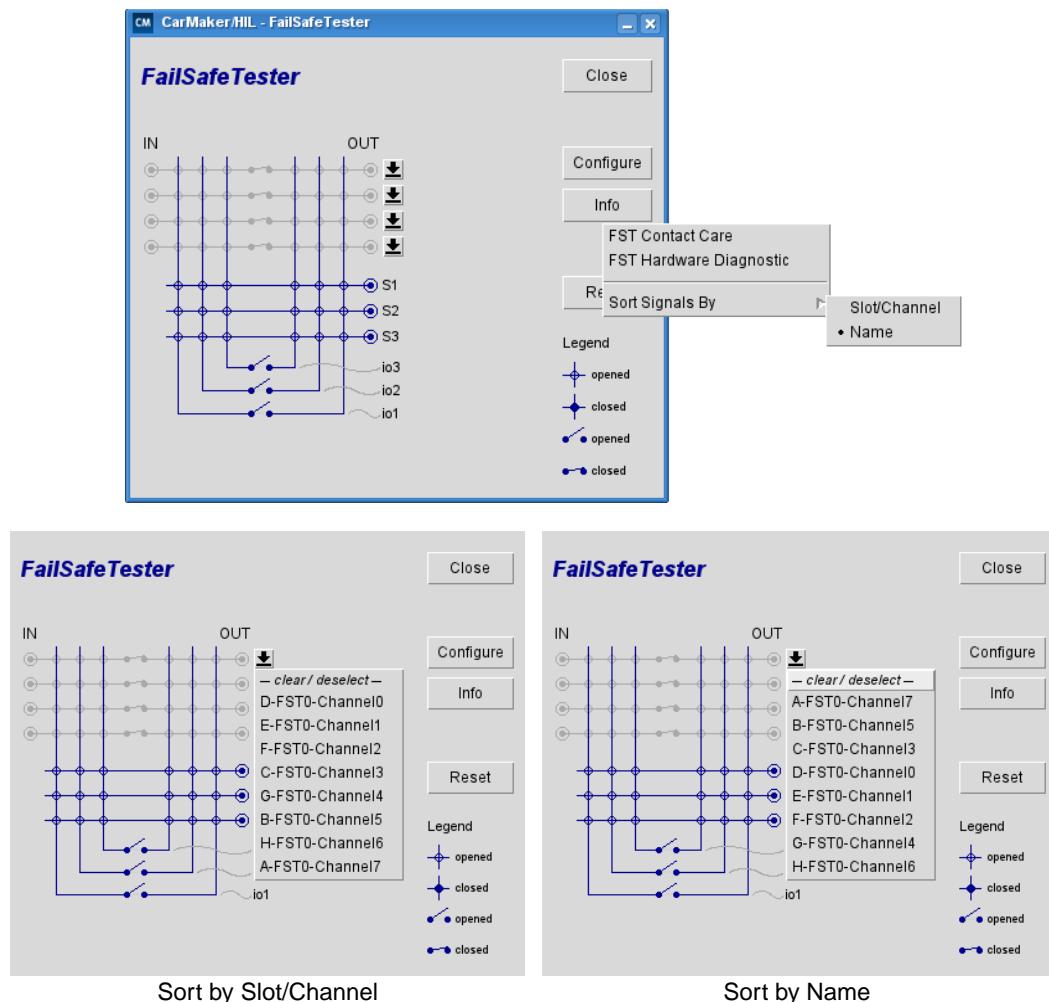
Figure 10.51: The FailSafeTester dialog for two FSTs

Due to its special connection matrix the FST-HVCs has an own frame within the FailSafeTester dialog.



All HVCs whos SUM point name is the same are available in the same HVC frame. The number of SUM points with different names determine how many FST-HVC frames can be shown.

With a huge number of configured signals it may be somewhat difficult to find the correct signal in the huge list of signal names. Therefore it is now possible to sort the list of signal names either by name or - as before - by slot and channel. The selection is done via the context menu on the right side of the FailSafeTester GUI.



The FailSafeTester dialog may look a little different depending on what cards are configured in your system. If, for example, there is no resistor card installed or configured the resistor fields will not be shown.



It is now possible to dynamically change the signal rows, the number of HVC frames and the number of HVC signal rows. This can be done in the context menu which is available by right-clicking inside FST frame. The number number of HVC signal rows additionally can be changed by clicking the "+" sign and "-" sign in the upper left corner of the HVC frame.

## Signals and InterConnection Lines

Figure 10.52 shows the same FailSafeTester dialog after being compartmentalized. Please refer to it during this section.

- The dialog is laid out as a matrix. The first eight horizontal rows contain the signals that should be manipulated. The red box (labeled A) shows the corresponding section of the dialog.



The number of signals that are displayed can be changed by setting the global variable `FST:::Cfg` in the `.CarMaker.tcl` file to a larger or smaller value. The command is:

```
set FST:::Cfg(nRls) <number of signals displayed>
```

The `.CarMaker.tcl` file is a hidden configuration file for the CarMaker GUI, located at top level of your CarMaker project folder. It can be edited with any text editor.

- The magenta box (labeled A.1) shows a single row inside this section that represents one signal that is wired through the FailSafeTester hardware. By selecting the drop down menu button at the right end of the row a list a selection list of all the available signals will be shown.

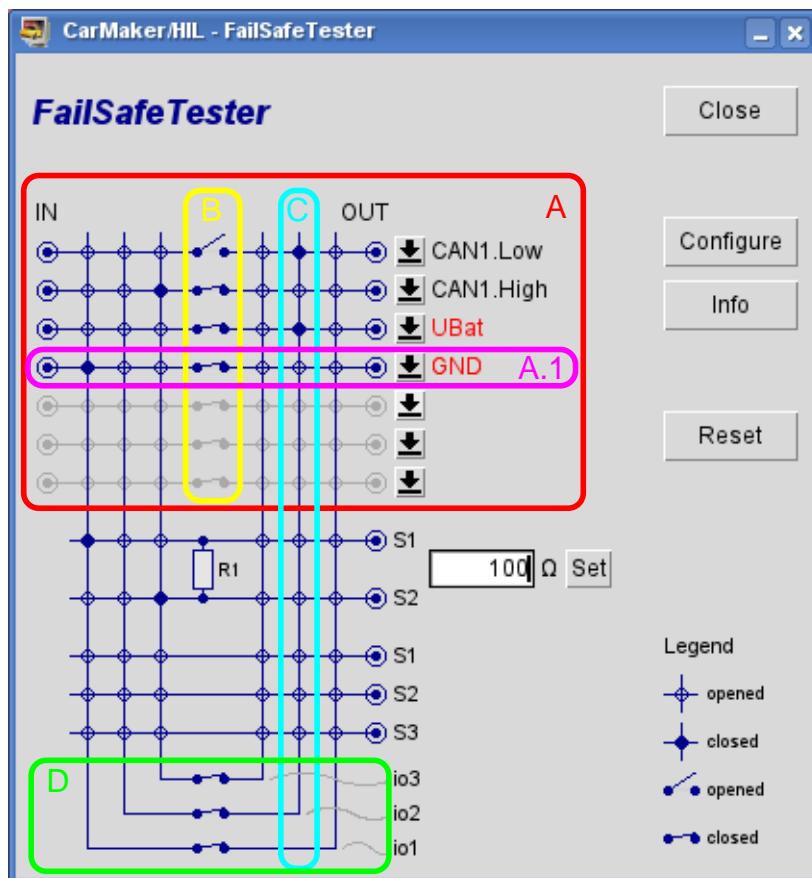


Figure 10.52: The Parts of the FailSafeTester Dialog

- The yellow box (labeled B) contains the SignalCut relays. They are used to cut and reconnect the signal that is selected in the respective row. Click the button to open and close the relay will open or close the connection. By default, these relays are closed (button shows picture of a horizontal line) but when selected the relay will open (bottom shows picture of broken line).
- The cyan box (labeled C) represents the InterConnection Lines (ILs) that are described in [section A.1.2 'InterConnection Lines'](#). The box shows one of the six possible ILs available with the standard relay cards (only 4 InterConnection Lines are available for HCRCs): Interconnection Line O2. The 'O' stands for Output and the '2' signifies that it is the second Output IL. The term 'Output' usually refers to the side that is connected to the ECU. By clicking the cross button at the juncture of an InterConnection Line and a signal, that relay will be closed and a connection between the two will be made. A blue dot over the cross represents a connection.
- The green box (labeled D) shows what could be termed an IL-Pairing Relay (InterConnection Line Pairing Relay). It simply connects or disconnects an input side InterConnection Line with its corresponding output side InterConnection Line. So, I1 can be

connected to O1, I2 can be connected to O2, and I3 can be connected to O3. Then, the six InterConnection Lines available with the Standard Relay Cards (only four InterConnection Lines available with High Current Relay Cards) will effectively become three, since the input and output side lines are paired together (thus the IL Pairing Relay name).

## FailSafeTester Dialog Example 1

Let's have an example. [Figure 10.53](#) shows the FailSafeTester dialog with two signals selected: CAN.Low and UBat.



This is a dangerous signal combination! The example illustrates the need to be careful with the signals that are connected to ensure that no damage is done to any of the hardware connected to the FailSafeTester, or the FailSafeTester itself.

- We can see from the picture that the first signal selected, CAN.Low, has two changes made. The first is that the SignalCut relay has been opened, thereby cutting the signal between the input (real time) side and the output (ECU side). Second, a connection has been made to the InterConnection Line O3.
- The second signal selected, UBat, has one change made to it. The signal has been connected to InterConnection Line O3, the same InterConnection Line that CAN.Low is connected to.

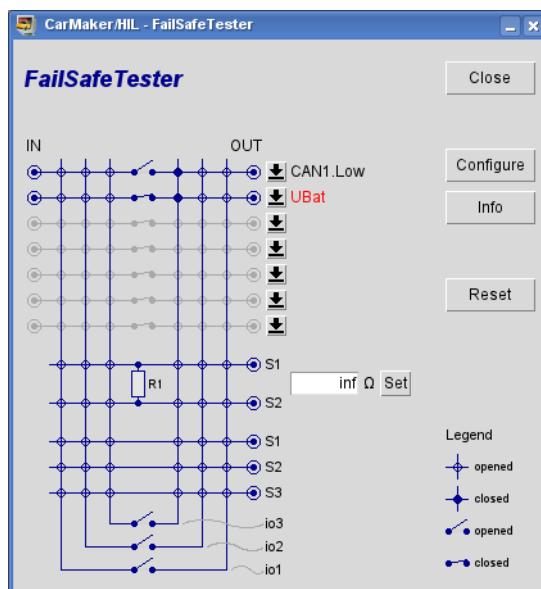


Figure 10.53: FailSafeTester Example 1

As a result, the CAN.Low signal going to the ECU is replaced with UBat (12V). This is, as you might expect, probably not a very good idea!

## Adding Resistors and Reading Signals

### Resistors

You can add a resistor to a signal or between two or more signals. When the resistance is added to one signal the effect could simulate corrosion or a similar decrease in the conductive properties of the wire. A resistor added between two or more signals would simulate current leakage, caused by faulty insulation, water, etc. The red box (labeled A) shows the area in the dialog that is used to set and connect the resistor.

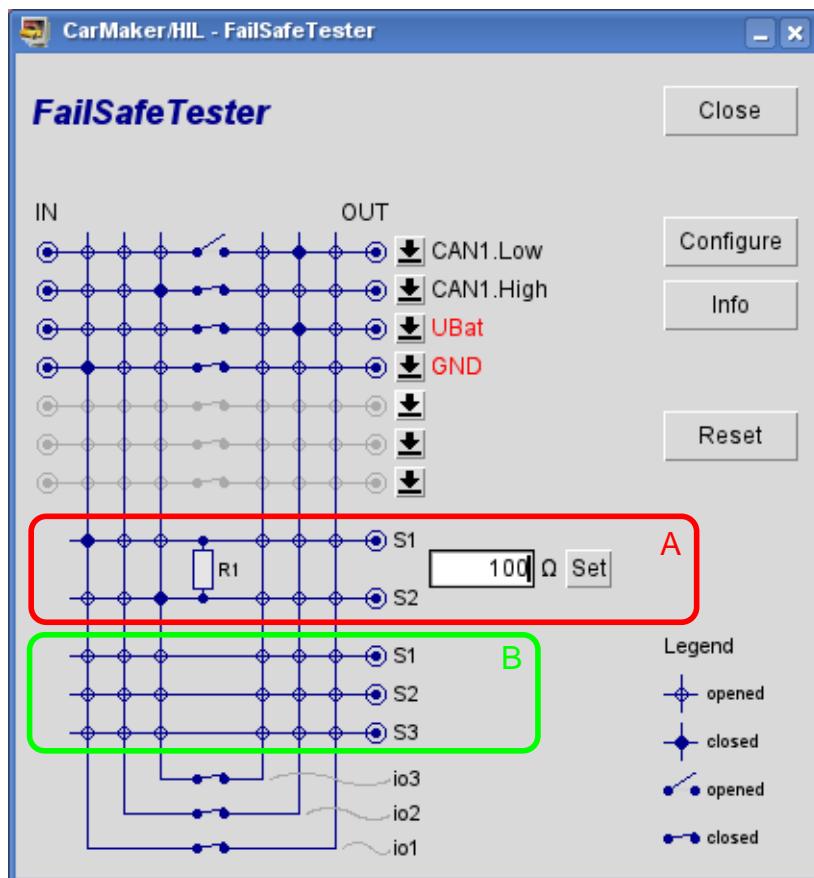


Figure 10.54: Resistors and Signal Readings

- To set the resistor, enter the desired resistance value in the entry field and press the [Set] button.
- S1 is one end of the resistor and S2 is the other end.
- For adding a resistor inside one signal (e.g. increased contact resistance) do the following:
  - define the resistor value
  - connect the signal to the S1 and S2 rows of the resistor with two InterConnection Lines on different sides of the SignalCut relay
  - open the appropriate SignalCut Relay



Pay attention on the order described above. If, for example, the SignalCut Relay is cut before the resistor is added to the signal the signal is cut (infinite resistance) until the resistor is added.

- To add a resistor between two signals (e.g. leakage current by defective insulation):
  - define the resistor value
  - connect S1/S2 of the resistor with two different InterConnection Lines.

- connect the two signals with the InterConnection Lines



When connecting S1/S2 to the corresponding InterConnection lines (I1/O1, I2/O2, I3/O3) take care that the pairing relays are not closed. If they are closed you will generate a short-circuit instead of a leakage current!

### Measuring Signals on the InterConnection Lines

Sometimes it is helpful to measure the signals with an external device such as an oscilloscope or multimeter. This is possible by connecting a SignalConnect Channel (shown in the green box labeled B in [Figure 10.54](#)) to a signal with an InterConnection Line, and plugging the measuring device into the corresponding banana jack at the front of the ControllerCard. If, for example, you have an oscilloscope plugged into the S1 jack on your ControllerCard and you would like to read the signal “wheel speed front left”, you can simply connect the wheel speed front left signal to an InterConnection Line, connect the S1 SignalConnect Channel to the same InterConnection Line, and see the signal displayed on your scope.

It is also possible to feed signals in, for example using a waveform generator, by going through a similar process. However, care should be taken when doing this, as it is possible to damage electronic components if a signal is fed to a module that is not designed to handle it.

### FailSafeTester Dialog Example 2

[Figure 10.56](#) inserts a resistor of 100Ohms to the signal ‘ WhISpd.FL.High’. Via the connector ‘S1’ and ‘S2’ on the FST-ControllerCard the signal level against GND can be measured with an external device.

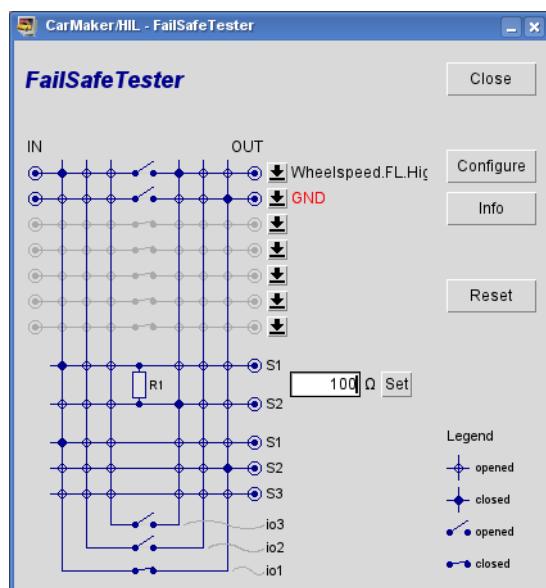


Figure 10.56: Resistor and SignalConnect example

## A.6.2 FailSafeTester Mini-Maneuvers Commands

The idea of the FailSafeTester is to be able to modify the signals that are wired through it in several ways. The modifications should only take effect to the signals that are selected for a particular test, and the effects should only occur when specified. In other words, we want to be able to define the signal(s) to be used, the action taken and the exact trigger condition for reproducible tests.

The first approach to do this is using FailSafeTester dialog as described before. The disadvantage is, that interactive actions done by the user are not fully reproducible.

The other approach is to do things automatically during a CarMaker simulation. This is done by adding commands to the mini maneuvers that define a TestRun. This section explains how to use the Mini Maneuver Command Language to control the FailSafeTester.

Please find further information about the Minimaneuver Command Language in general in [section 'Minimaneuver Command Language' on page 590](#) and a list of all FailSafeTester specific commands in [section B.1.7 'FailSafeTester Commands' on page 599](#).

### Groups, InterConnection Lines and Signals

We have already discussed InterConnection lines and signals in [section A.1.2 'InterConnection Lines'](#). Remember that an InterConnection line is used to connect the signals with other signals, resistors, or with external devices (e.g. multimeters and oscilloscopes). Normally there are six available InterConnection lines, but in the case of High Current Relay cards there are only four. Half of the InterConnection Lines are located on one side of the SignalCut Relays, and the other half of the InterConnection Lines are located on the opposite side of the SignalCut Relays. The InterConnection Lines can be paired by connecting them with IL-Pairing Relays, which simply join the InterConection Lines on one side with its corresponding InterConnection Line on the other side. [Figure 10.57](#) presents this information in graphical form.

In contrast to the FailSafeTester dialog, the IL-Pairing relays are always closed when using the mini maneuver command language. It is not possible to open them. As shown in [Figure 10.57](#) the 6 available InterConnection lines (4 available with High Current Relay Cards) will effectively become three (two for High Current Relay Cards).

For using the FailSafeTester with Mini-Maneuvers a high-level representation is used called a group. All participants of a group are connected to each other. This is realized by choosing an InterConnectionLine of the FST and make the appropriate connections. Due to the limited number of InterConnectionLines only three simultaneous groups are possible with Standard Relay Cards (6 InterConnection Lines divided by 2) and only 2 simultaneous groups are possible with High Current Relay Cards (4 InterConnection Lines divided by 2).

### Adding Signals, Resistors, and SignalConnection Channels to a Group

A group can consist of a number of signals, resistors (each side of a resistor is connected to one group) and SignalConnection channels (connect the FST to external devices). Groups are distinguished by their names.

The names:

- must start with a letter (alpha character)
- can contain both letters and numbers (alpha-numeric)
- should not contain any punctuation marks, special characters, etc.
- can be a maximum of 60 characters long

#### Add Signals to a Group

FSTChAd <Grp> <Ch1>,<Ch2>,...

A signal is added to the group by the command `FSTChAd`. The first argument of this command is the name of the group. If the group does not exist it will be created. The second argument is a comma separated list of channels (a channel in this sense is synonymous to a signal or a resistor). When added to a group, a channel is connected to all the other channels that are in that group (e.g. that were previously or simultaneously added).

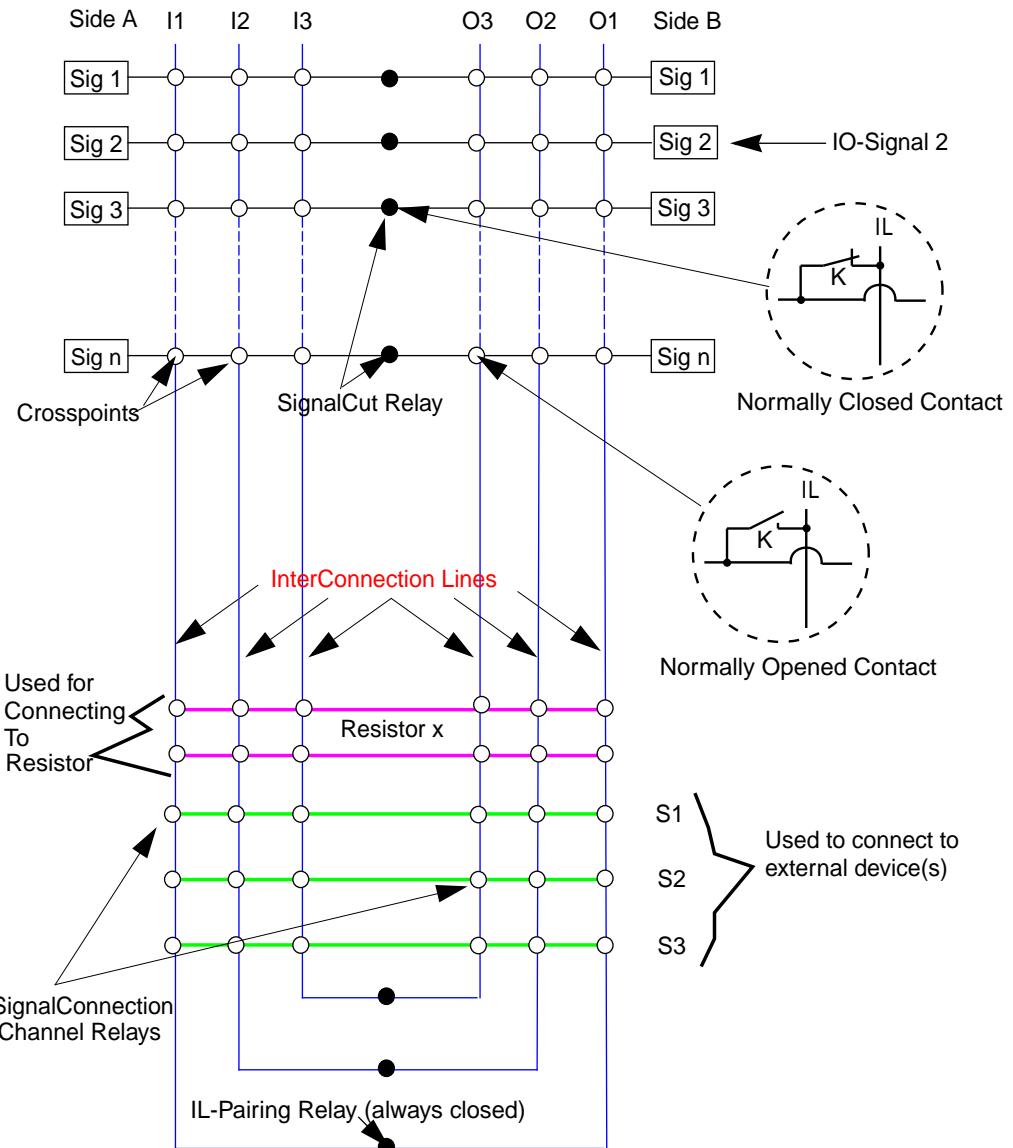


Figure 10.57: The FailSafeTester Matrix

### Inserting Resistors

`FSTR <RstName> <Value>`

Using this command, a new resistor with specified name and values is generated. To add it to an existing group, use the `FSTChAd` command with the group name as first argument and the resistor name and side (separated by a dot) as second argument. Please refer to section B.1.8 'Resistors - Naming conventions' on page 601.

## Adding SignalConnection Channels to a Group

```
FSTSCAd <SCName> <Grp>
```

The command, FSTSCAd takes as its first arguments the name of a SignalConnection Channel. The three possibilities (S1, S2, and S3) correspond to the three female banana jacks located on the front of the ControllerCard. The second argument is the name of a group. If the group does not exist when this command is called, the group will be created. When used, the command will connect the specified SignalConnection Channel to the specified group.

## Removing channels from a Group

To remove a channel from a group, the following command is used:

```
FSTChRm <Grp> <Ch1>,<Ch2>,...
```

The first argument is the group, and the second is a comma separated string that contains the names of the channels that are to be removed.

## Cutting and Connecting Signals

Cutting a signal (e.g. opening the SignalCut Relay) is done by using the command:

```
FSTChCt <Sig1>,<Sig2>,...
```

The argument is a comma separated list of the signals that should be cut.

Reconnecting a signal (e.g. closing the SignalCut Relay) is done by using the command:

```
FSTChCn <Ch1>,<Ch2>,...
```

The argument is a comma separated list of the signals that should be connected.

An Overview over all FailSafeTester commands can be found in [section B.1.7 'FailSafeTester Commands' on page 599](#).



You can find examples in the TestRuns under Examples > FailSafeTests.

## FailSafeTester Example 1

### The Scenario

Let's examine a hypothetical situation. Suppose the intention of your TestRun is to determine what happens if an ESP does not receive the wheel speed signals of the front wheels during a breaking maneuver on a road with a low coefficient of friction.

Your motivation is to consider if your new algorithm designed to handle this situation works correctly. Therefore, the front wheel speed signals must be cut moments before the brake pedal is pressed. How to do that?

## The Solution

### Step 1 - Define a TestRun

Define a TestRun with the desired road conditions, tires, etc. that also contains two maneuver steps. The first is used to accelerate the car to the desired velocity. The second is the actual braking maneuver.

### Step 2 - Add the Command

In the second maneuver step, add a mini maneuver command that will cut both front wheel speed signals after one second. Use the following command:

```
[Time>1] FSTChCt WhlSpd.FL, WhlSpd.FR
```

### Step 3 - Simulate

Run the simulation. One second after starting the braking maneuver the front wheel speed signals will be cut.

## Appendix B

# Minimaneuver Command Language

## B.1 Command Reference

The Minimaneuver Command Language gives the user the possibility to define maneuver actions in a very flexible manner. Apart from all Realtime Expression operators a number of special minimaneuver commands are available. This chapter lists all these special minimaneuver commands.

Please find more information on the whole command syntax including trigger definitions in [section 4.5.8 'Minimaneuver Command Language' on page 107](#)

### B.1.1 Driving Maneuver Commands

With the driving maneuver commands it is possible to influence the standard top down execution order of minimaneuvers. Additionally it is possible to manipulate the *External Inputs From File* functionality.

#### DMjmp - Driving Maneuver Jump

DMjmp      *LabelStr/ManNo*  
DMjmpn     *ManNo*

Jump to the minimaneuver with the label *<LabelStr>*. The label has to be set in the desired minimaneuver by entering a name in the field *Label* (see [section 'Label' on page 93](#)). Alternatively, the maneuver number *<ManNo>* of the minimaneuver can be used to specify the target maneuver.

**Example**      Inserting a DrivMan Jump in a braking test:  
In the first maneuver step "Accelerating" jump after 20s directly to the braking maneuver (step 3) and miss out step 2 "Free Rolling".  
In the first maneuver step add the command: **[Time>=20] DMjmpn 2.**

Note that the maneuver numbering starts at zero!

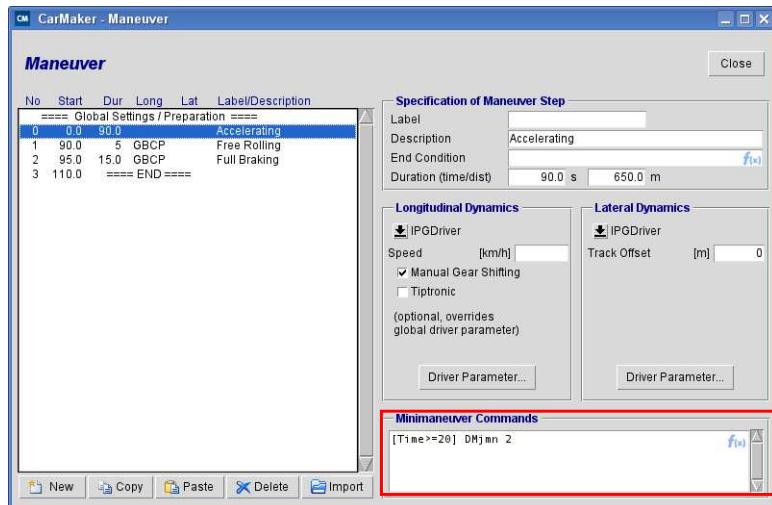


Figure 10.58: Inserting a maneuver jump

### Example

Defining a label called "Brake" in the last maneuver step, it can be used to address the maneuver jump: **[Time>=20] DMjmp Brake.**

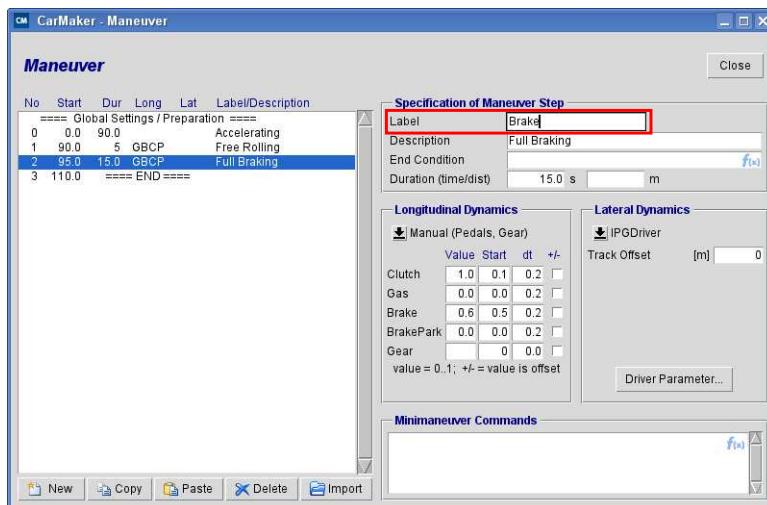


Figure 10.59: Inserting a label to address the maneuver jump

## Driving Maneuver External Inputs from File

`ExtInp file ModeStr`

Modify the CarMaker standard behavior of reading inputs from file directly from the beginning of a simulation. With this command it is possible to define the point when to start the input from file. This can either be at the beginning of a minimaneuver or in combination with a trigger condition somewhere during a minimaneuver.

The parameter *ModeStr* specifies the following actions:

ModeStr	Description
enable	Enable input from file. If this is the first instance of this command the input from file starts at this place.

ModeStr	Description
disable	Disable input from file. This can be used as an interrupt function. The internal time used for input from file continues to count up but the values are not updated in CarMaker. Any further 'enable' commands make sure that the values are updated in CarMaker again.
restart	With this parameter it is possible to rewind the timeline of the input from file module. Behavior is like the first start of input from file.

To find more about the *Input from File* functionality please read [section 4.6 'Input From File / Using Real Measurements' on page 110](#).



Do not replace the key `file` by the name of the input file! The whole command string is `ExtInp` file `disable`!

**Example** Please find an example on using these file commands in the TestRun Examples/CarMakerFunctions/ExternalInputs/ActionFromFile-02. Use the following command to disable the input from an external file after 5s simulation time: `[Time]>=5] ExtInp file disable`.

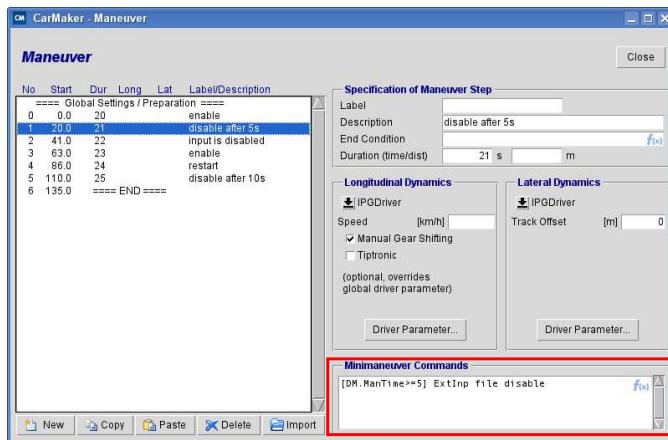


Figure 10.60: Disable the input from an external file after 5s

## B.1.2 Direct Variable Access Commands

DVA makes it possible to manipulate some quantities of the CarMaker data dictionary. Further information about the DVA interface can be found in [section 6.3 'DVA: Online Manipulation of the Simulation' on page 357](#). The following commands provide the DVA functionality within a minimaneuver:

### DVAwr - Direct Variable Write Access

`DVAwr <QuantName> <Mode> <nCycles> <Val1> [Val2] [nCyclesRamp]`

Table 10.10 List of arguments of the DVA command

Argument	Description
<code>&lt;QuantName&gt;</code>	Name of the quantity to access via DVA
<code>&lt;Mode&gt;</code>	Abs, Off, Fac, FacOff, AbsRamp, OffRamp, FacRamp, FacOffRamp
<code>&lt;nCycles&gt;</code>	Number of cycles the selected quantity should be overwritten, set -1 for a permanent action
<code>&lt;Val1&gt;</code>	Value to be used for DVA
<code>&lt;Val2&gt;</code>	Used to specify offset or ramp in some modes
<code>&lt;nCyclesRamp&gt;</code>	Used to specify number of cycles used for the ramp

**QuantName** The first argument defines the quantity to be overwritten. By clicking with the right mouse button in the Minimaneuver Command Language section of the Maneuver dialog, a selection list will appear with all User Accessible Quantities. However, not all quantities can be influenced by a DVA command, some quantities are read-only. The quantities which can be overwritten by DVA are highlighted in the selective list.

**Mode** This command can be used to overwrite the current value of a quantity via DVA. Several modes are possible:

Table 10.11 Possible DVA modes

Mode	Description
Abs	<Val1> specifies the absolute value to overwrite the quantity.
Off	<Val1> specifies an offset which is added to the current value of the quantity.
Fac	<Val1> specifies a factor the current value of the quantity is multiplied with.
FacOff	Combination of the cases 'factor' and 'offset' above. <Val1> specifies the factor, <Val2> specifies the offset.
AbsRamp	Specifies absolute value in <Val1> and the number of cycles <nCyclesRamp> to be used to fade to this value.
OffRamp	Specifies the offset value to be added to the current value of the selected quantity in <Val1> and the number of cycles <nCyclesRamp> to be used to reach the given offset.
FacRamp	Specifies the factor the current value of the selected quantity is multiplied with in <Val1> and the number of cycles <nCyclesRamp> to be used to reach the given factor.
FacOffRamp	Specifies the factor the current value of the selected quantity is multiplied with in <Val1>, the offset value to be added to the current value of the selected quantity in <Val2> and the number of cycles <nCyclesRamp> to be used to reach the given factor and offset.

**nCycles** The number of cycles <nCycles> determines how long the quantity should be overwritten. If a mode of type *Ramp* is specified <nCyclesRamp> can be larger than <nCycles>. The command is always aborted once the number of cycles specified in <nCycles> is reached - even if the target value for the ramp has not been achieved yet.



It should be noted that modes applying an offset or a factor to a quantity only works as expected if the quantity is reset/recalculated during each cycle. Otherwise an unexpected behavior may result, e.g. exponential growth of a quantity if a factor is applied.



Entering -1 for <nCycles> means, the action has no timely limitation. Thus, the DVA write command is valid until a DVA reset command follows or the whole CarMaker application is shut down. The quantity is not reset at the end of the current simulation!

When using -1 as duration, you should always use a final maneuver to reset all DVA commands (see [section • 'Final Maneuver If the last minimaneuver has a duration of zero seconds, it has a special meaning: This maneuver is called at the end of the TestRun even if the TestRun was stopped ahead of time \(by user, by an error, ...\). The actions defined here will be conducted at any rate.'](#) on page 99). The command is called **DVArel - DVA release**.



Please note: As opposed to former CarMaker versions (<4.0>), the DVA access point does not have to be defined in the dialog. Each quantity which can be overwritten using DVA knows its proper interface point.



As an example take the TestRun Examples > VehicleDynamics > Rollover > MovingMass.

### DVArel - DVA release

`DVArel <QuantName>`

Release the specified quantity.

`DVArel *`

Release all quantities overwritten.

The following example will impose two steer steps via DVA. After 15s the quantity `DM.Steer.Ang` will be overwritten by the absolute value of 0.349 rad. The command will last 1000 cycles which means at realtime 1s. After another 10 seconds another steer angle pulse of 0.436 rad will be applied. The command line would look as follows:

**Example**    [Time>= 15] DVAwr DM.Steer.Ang Abs 1000 0.349  
               [Time>=25] DVAwr DM.Steer.Ang Abs 1000 0.436

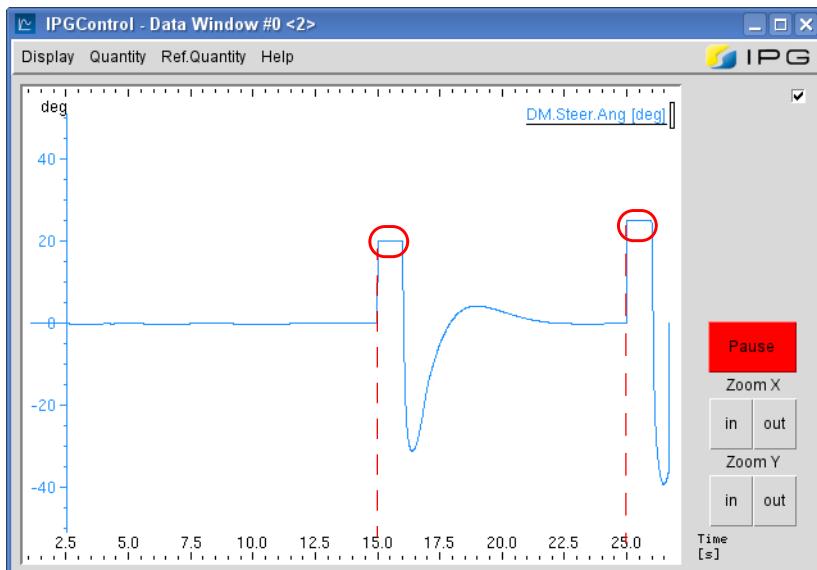


Figure 10.61: DVA command to manipulate the steering wheel angle. See the DVA inputs highlighted.



For this see also the example TestRun Examples > CarMakerFunctions > DVA > Straight\_DVA



A more convenient way to overwrite signals might be the usage of Realtime Expressions. The steering wheel angle example from above could be implemented with Realtime Expressions using an If-Then construct. Realtime Expressions are always executed after the Driveman module, ahead of the DVA command with location "DM". Please note that all Realtime Expressions start with the prefix `Eval` in the minimaneuver command language:

**Example**    `Eval Time>=15 && Time<=16 ? DM.Steer.Ang=0.349`  
               `Eval Time>=25 && Time<=26 ? DM.Steer.Ang=0.43`

## B.1.3 Extended RealtimeExpressions Commands

You are not limited to use Realtime Expressions as trigger for minimaneuver commands only. There are some extended functionalities such as creating new quantities, temporary variables or recording some signals between two events etc. If you want to use a Realtime Expression as a minimaneuver command and not just as a trigger, the syntax is as follows:

```
Eval <RTExpCmStr>
```

**Example** The Realtime Expression command string is indicated by the prefix *Eval*. The Realtime Expression command string can be made of any function explained in [appendix 'Realtime Expressions' on page 602](#).

## B.1.4 Action Commands

Beside the modification of CarMaker quantities via DVA it is possible to perform certain pervasive actions with the following special commands.

### VhclOp

```
VhclOp=<Command>
```

Set a target operation state of the vehicle operator or enable/disable the vehicle operator depending on the command *Command*. Following commands are supported:

Command	Description
<b>disable</b>	Disables the vehicle operator
<b>enable</b>	Enables the vehicle operator with the previous target operation state
<b>absent</b>	Enables the vehicle operator with the target operation state "Absent"
<b>poweroff</b>	Enables the vehicle operator with the target operation state "Power Off"
<b>poweracc</b>	Enables the vehicle operator with the target operation state "Power Accessory"
<b>poweron</b>	Enables the vehicle operator with the target operation state "Power On"
<b>drive</b>	Enables the vehicle operator with the target operation state "Drive"
<b>user&lt;i&gt;</b>	Enables the vehicle operator with any user defined target operation state "User<i>"

**Example** VhclOp=drive

### Key

```
Key=<KeyPos>
```

Set the vehicle key to the position *KeyPos*. Following key positions are supported:

Position	Description
<b>out</b>	Vehicle key position "Outside"
<b>off</b>	Vehicle key position "Power Off"
<b>acc</b>	Vehicle key position "Power Accessory"
<b>on</b>	Vehicle key position "Power On"
<b>starter</b>	Vehicle key position "Starter"

**Example** Key=on

## SST

SST=<0/1>

Switch SST (vehicle start-stop button) on or off.

**Example** SST=1 (Start-stop button on)

**Example** SST=0 (Start-stop button off)

## KI15

KI15=<0/1>

Switch KI15 (ignition) on or off.

This command was retained for compatibility to CarMaker version 4.5 with slightly modified functionality.

If the vehicle operator is activated, the command sets the target operation state to "Power On" or "Power Off", otherwise the vehicle key is set to the position "Power On" or "Power Off".

**Example** KI15=1 (Switch ignition on)

**Example** KI15=0 (Switch ignition off)

## KI50

KI50=<0/1>

Switch KI50 (StarterControl) on or off.

This command was retained for compatibility to CarMaker version 4.5 with slightly modified functionality.

If set to 1, the vehicle key is set to the position "Starter", otherwise the vehicle key is set to the position "Power On".

**Example** KI50=1 (Switch StarterControl on)

**Example** KI50=0 (Switch StarterControl off)

## Engine

Engine=<0/1>

Switch Engine on or off. To be used for vehicles without engine starter (KI50). Activates the engine starter internally.

This command was retained for compatibility to CarMaker version 4.5 with slightly modified functionality.

If set to 1, the vehicle start-stop button SST is set to 1.

**Example** Engine=1 (Switch EngineSwitch on)

**Example** Engine=0 (Switch EngineSwitch off)



If in the first minimaneuver one of the commands above (SST, Engine, KI15, etc.) is specified, the automatic start of the powertrain is disabled by deactivating the vehicle operator in the preparation phase. The user is responsible himself for turning on the vehicle by activating the vehicle operator or by switching the key, start-stop button SST and the pedals manually.



As an example take the TestRun Examples > CarMakerFunctions > ManeuverControl > ManualEngineStart.

## B.1.5 Logging Commands

The minimaneuver command language provides basic functionality to issue messages to the CarMaker session log (CarMaker GUI > Simulation > Session Log).

### Log

```
Log      "<Msg>"  
LogWrnS "<Msg>"  
LogErrS "<Msg>"
```

The command outputs the message <Msg> (string embedded in double quotes) to the session log. The following types of messages are distinguished:

Table 10.12 Logging command kinds

Command	Description
Log	Issue a log message. The message is specified in [Msg].
LogWrnS	Issue a warning message. The message is specified in [Msg].
LogErrS	Issue an error message. The message is specified in [Msg].

**Example** Log "This is my own logging message"

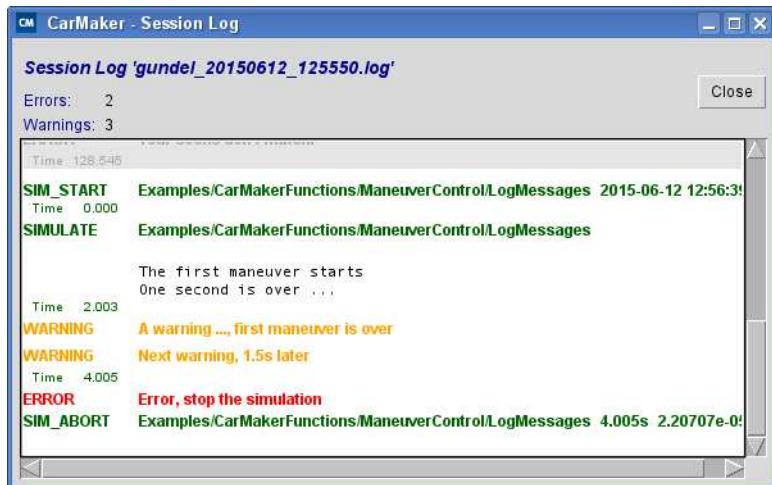


Figure 10.62: Log message which appears in the CarMaker Session Log window



As an example take the TestRun Examples > CarMakerFunctions > ManeuverControl > LogMessages.

## B.1.6 Data Storage Commands

### DStoreSave

`DStoreSave TimeHistory TimeFuture`

Starts saving to data storage. Store buffered data for *TimeHistory* seconds and for the following *TimeFuture* seconds.

Table 10.13 Data storage options

Abbreviation	Description
THistory	Defines the time span of the data already being in the buffer which should be saved.
TFuture	Defines the time span during which the results should be saved: TFuture = -1: unlimited storing from this point on TFuture > 0: user specific duration from this point on TFuture = 0: end storing from this point on

The unit for all time parameters is s (seconds).



### DStoreStop

`DStoreStop`

Stops saving to the data storage.

### DStoreAbort

`DStoreAbort`

To abort the data storing. The result file is not kept, it is removed.



As an example take the TestRun Examples > CarMakerFunctions > ManeuverControl > DStore.

## B.1.7 FailSafeTester Commands

Using a FailSafeTester in combination with CarMaker/HIL offers great possibilities to evaluate the controller behavior in all situations. Electrical faults can be inserted by triggering relays on the FailSafeTester card. The failure insertion can be easily activated manually by using the FailSafeTester dialog in the CarMaker/HIL GUI. However, this way of manually triggering failures is hardly reproducible, so several minimaneuver commands are available to execute fail safe tests fully automated and reproducible. Below, all possible FailSafeTester commands are listed. Please find more general information on the FailSafeTester in appendix '[FailSafeTester with CarMaker/HIL](#)' on page 558.

### FSTInt - Initialize FST

`FSTInt`

Before any other FailSafeTester commands are executed the FailSafeTester needs to be initialized (usually at the beginning of a simulation). Please find more information in the FailSafeTester documentation, section INIT instruction.

### FSTRst - Reset FST

`FSTRst`

Resets all FST-cards to the default state. Please find more information in the FailSafeTester documentation, section RESET instruction.

### FSTPng - Send Ping to FST

`FSTPng`

Sends a ping to the FailSafeTester and logs the answer to the session log. Please find more information in the FailSafeTester documentation, section PING instruction.

### FSTChAd - Add Channels to a Group

`FSTChAd <Grp> <Ch1>, <Ch2>, ...`

Adds the selected channels to a group. All channels of the group will be connected. If the group does not exist it will be created. If the maximum number of groups is exceeded an error message is issued in the session log.

Table 10.14: Arguments of the Add Channel command

Argument	Description
<code>&lt;Grp&gt;</code>	Name of the group.
<code>&lt;Ch1&gt;, &lt;Ch2&gt;, ...</code>	List of channel names.

### FSTSCAd - Add Signal Connector

`FSTSCAd <SCName> <Grp>`

Adds a signal connector on the controller board to a group. This command behaves like the *Add Channel* command except that it refers to external signal connector plugs.

Table 10.15: Options of the Add Signal Connector command

Argument	Description
<code>&lt;SCName&gt;</code>	Name of the signal connector (S1, S2, S3) to join the group.
<code>&lt;Grp&gt;</code>	Name of the group.

## FSTChRm - Remove Channels from Group

`FSTChRm <Grp> <Ch1>, <Ch2>, ...`

Removes the selected channels from a group. The channels will be disconnected from the remaining channels. If a resistor signal is removed from a group all other group memberships (input and output signals of the resistor) are removed as well. The resistor keeps its resistor value, though.

Table 10.16: Options of the Remove Channels command

Argument	Description
<code>&lt;Grp&gt;</code>	Name of the group.
<code>&lt;Ch1&gt;, &lt;Ch2&gt;, ...</code>	List of channel names.

## FSTGrpRm - Remove Group

`FSTGrpRm <Grp>`

Remove all connections created by the group. Remove all channels from the group. Then remove the group itself.

Table 10.17: Options of the Remove Group command

Argument	Description
<code>&lt;Grp&gt;</code>	Name of the group or '*' to remove all groups.

## FSTChCt - Cut Channels

`FSTChCt <Ch1>, <Ch2>, ...`

Cuts the selected channels (signal cut).

Table 10.18: Options of the Cut Channels command

Argument	Description
<code>&lt;Ch1&gt;, &lt;Ch2&gt;, ...</code>	List of channel names.

## FSTChCn - Connect Channels

`FSTChCn <Ch1>, <Ch2>, ...`

(Re-)Connects the channels.

Table 10.19: Options of the Connect Channels command

Argument	Description
<code>&lt;Ch1&gt;, &lt;Ch2&gt;, ...</code>	List of channel names or '*' to connect all channels.

## FSTR - Set Resistor Value

`FSTR <RstName> <Value>`

Sets a resistor value in [Ohm].

Table 10.20: Options of the Set Resistor Value command

Argument	Description
<code>&lt;RstName&gt;</code>	Name of the resistor.
<code>&lt;Value&gt;</code>	Set resistor value (Ohm).

## FSTLGrp - List Group

**FSTLGrp**

List all groups and group members to the session log.

## FSTLFST - List Fail Safe Tests

**FSTLFST**

Show relay positions. Output is written to the session log.

## B.1.8 Resistors - Naming conventions

To specify the side of the desired resistor, use the <resistor name>.S1, <resistor name>.S2 naming convention. For example, if the resistor is named R1 then use the name R1.S1 to refer to the S1 side of the resistor. Similarly, use R1.S2 to refer to the other side of the resistor.

**Example**     The following command adds the S2 side of resistor R1 to the group g0:  
**FSTChAd g0 R1.S2**

## B.1.9 Example of FailSafeTester Commands

The following example shows how FailSafeTester commands can be added to a TestRun to control the FailSafeTester.

The Log command can be used to write comments to the log file (see [section B.1.5 'Logging Commands' on page 597](#)).

```

1: FSTRst
2: FSTChCt WhlSpd.FL
3: FSTChCt WhlSpd.FL,UBat,DA.abc
4: FSTChCt *
5: Log "/// Cut all channels"
6: FSTChCn *
7: Log "===" Connect all channels"
8: FSTGrpRm g0,g1,g2
9:
10: FSTGrpRm *; Log "free all groups"
11:
12: FSTChAd g0 WhlSpd.FL/o,UBat
13: Log "add out side of WhlSpd.FL and UBat"
14: FSTChAd g0 WhlSpd.FL/i,UBat
15: Log "add in side of WhlSpd.FL and UBat"
16: FSTChAd g0 WhlSpd.FR,Ground
17: Log "connect chls, in/out n.spec."
18:
19: Dist=1000 FSTChRm g0 UBat
20:
21: FSTSC S1 g0
22: FSTSC S2 i1
23: FSTSC S3 o3
24: FSTSC S2 0
25: Log "Disconnect S2"
26:
27: Time=10 FSTR R0 120

```

Figure 10.63: Example command list to control the FailSafeTester

As the list of commands is quite long it can be useful to define a new ScriptControl procedure which includes all these action commands. The procedure needs to be written to a text file which can be loaded in the Maneuver dialog under Global Settings (see [section 'Start Values' on page 95](#)). Please find more information on the ScriptControl programming language in the Programmer's Guide.

## Appendix C

# Realtime Expressions

## C.1 Fields of Application

Realtime Expressions have a wide range of application areas. Wherever a condition or special command including math operators or the creation of new quantities is required which has to be evaluated in realtime, this CarMaker feature can be used. In the CarMaker GUI, the possible use of Realtime Expressions is indicated by a special icon - a blue functions symbol: 

Realtime Expressions can be used in the following positions in the CarMaker application:

- Maneuver definition: as end condition of a maneuver step (see [section 'End Condition' on page 93](#))
- Maneuver definition: as trigger in the minimaneuver command language (see [section 'Trigger' on page 108](#))
- Maneuver definition: as command string in the minimaneuver command language (see [section B.1.3 'Extended RealtimeExpressions Commands' on page 595](#))
- IPGTraffic: as trigger to maneuver steps of various traffic objects (see [section 4.113 'Positioning of a traffic object in bird's view' on page 133](#) and [section 'End Condition' on page 136](#))
- SimParameters: The entry *SessionCmds* allows a definition of commands to be executed permanently (including the idle state) using the simulation program, *TestrunCmds* allows a definition of commands to be executed during a specific simulation (see section 'SimParameters > Session TestRun Commands' in the ReferenceManual)
- IPGRoad: as marker type *DrivMan Cmd* (see [section 'DrivMan Cmd' on page 64](#))
- ScriptControl: e.g. as trigger for Scratchpad records (see Programmer's Guide).

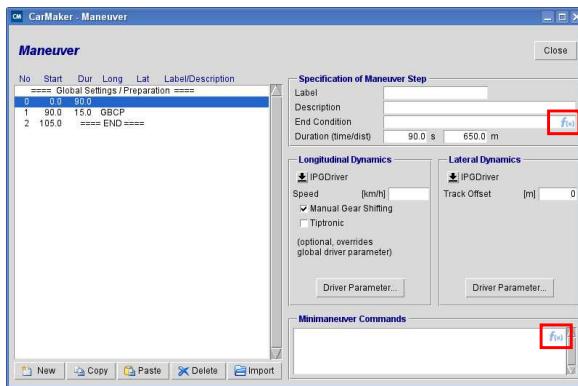


Figure 10.64: Realtime Expressions in the Maneuver dialog, indicated by the blue  $f(x)$ -icon

## C.2 Operators and Functions

The following pages list all possible math operators and functions used on Realtime Expressions.



Please note, that Realtime Expressions always use their default units! If you are not sure about the default unit of a quantity, please have a look at the Reference Manual, chapter "User Accessible Quantities".

### C.2.1 Arithmetic Operators

When combining more than one operator the statement with the highest priority is stated first, operators with same priority in the same row.

Table 10.21: Arithmetic operators for Realtime Expressions

Syntax	Description
$()$	<a href="#">Round Brackets</a> for grouping, influencing execution order
$-a$	Reverse sign of $a$
$a+b, a-b$	Add $a$ to $b$ , subtract $b$ from $a$
$a*b, a/b$	Multiply $a$ with $b$ , divide $a$ by $b$
$a \% b$	Modulo: floating point remainder of division $a/b$ (also valid for type double)
$a^{**}b$	$a$ to the $b$ power: $a^b$
$<<, >>$	Arithmetic bit-shift left/right
$a < b, a > b$	Comparison: $a$ less than $b$ , $a$ greater than $b$
$a \leq b, a \geq b$	Comparison: $a$ less or equal than $b$ , $a$ greater than or equal to $b$
$a == b, a != b$	$a$ equal $b$ , $a$ non-equal $b$
$a = b$	$a$ assigned to $b$ (associative to the right)
$a \& b$	Bitwise AND
$a ^ b$	Bitwise XOR
$a   b$	Bitwise OR
$a \&\& b$	Logical AND

Table 10.21: Arithmetic operators for Realtime Expressions

Syntax	Description
$a \wedge b$	Logical XOR
$a \parallel b$	Logical OR
$a+=b, a-=b$	Arithmetic assignment: $a$ is the sum/subtraction of $a$ and $b$ ( $a=a+b; a=a-b$ )
$a*=b, a/=b$	Arithmetic assignment: $a$ is a multiplied/divided by $b$ ( $a=a*b; a=a/b$ )
$a\%b, a^{**}b$	Arithmetic assignment: $a$ is a modulo/to the power $b$ ( $a=a\%b; a=a^{**}b$ )
$a\&=b, a^{\wedge}=b, a {=}b$	Arithmetic assignment: $a$ is bitwise AND/XOR/OR conjunction of $a$ and $b$ ( $a=a\&b; a=a^{\wedge}b; a=a {=}b$ )
$a\&&=b, a  =b$	Arithmetic assignment: $a$ is logical AND/OR conjunction of $a$ and $b$ ( $a=a\&b; a=a  b$ )
$a ? b : c$	<a href="#">Conditional Expression</a> : if $a$ then $b$ else $c$ (associative to the right)
<code>cmd1; cmd2</code>	<a href="#">Sequence</a> : list more than one command in one line

## C.2.2 Built-in Functions

Table 10.22: Realtime Expressions built-in functions

Function	Description
<code>int(a)</code>	Returns integer of $a$
<code>sgn(a)</code>	Returns 1 if $a$ is positive, -1 if $a$ is negative, 0 if $a == 0$
<code>copysign(a,b)</code>	Returns value of $a$ with sign of $b$
<code>random()</code>	Creates random value between 0 and 1 (float)
<code>random(a,b)</code>	Creates random value between $a$ and $b$ (float)
<code>bool(a)</code>	Returns 1 if $a$ is true, returns 0 if $a$ is false
<code>abs(a)</code>	Absolute value of $a$
<a href="#"><code>ceil(a)</code></a>	Returns the smallest integral value that is not less than $a$ (equal to $a$ if $a$ is integer)
<a href="#"><code>floor(a)</code></a>	Returns the largest integral value that is not greater than $a$ (equal to $a$ if $a$ is integer)
<a href="#"><code>trunc(a)</code></a>	Round $a$ to the nearest integer not larger in absolute value
<code>fract(a)</code>	$a-\text{trunc}(a)$
<code>min(a,b), min(a,b,c), min(a,b,c,d)</code>	Find minimum of given arguments
<code>max(a,b), max(a,b,c), max(a,b,c,d)</code>	Find maximum of given arguments
<code>clamp(a,b,c)</code>	Clamping arguments: $(a < b) ? b : (a > c) ? c : a$ if $a$ less $b$ return $b$ else: if $a$ greater $c$ return $c$ else: return $a$
<code>mix(a,b,c)</code>	Mixing of arguments: $a*(1-c)+b*c$ ( $c [0..1]$ blinks from $a$ to $b$ )
<code>n++</code>	Increments the value of $n$ and returns the original value of $n$ held before incrementing (postfix increment)
<code>n--</code>	Decrements the value of $n$ and returns the original value of $n$ held before decrementing postfix decrement

Table 10.22: Realtime Expressions built-in functions

Function	Description
<code>++n</code>	Increments the value of <i>n</i> and returns the incremented value of <i>n</i> (prefix increment)
<code>--n</code>	Decrements the value of <i>n</i> and returns the decremented value of <i>n</i> (prefix decrement)
<code>sin(a), cos(a), tan(a)</code>	Trigonometrical functions
<code>asin(a), acos(a), atan(a)</code>	Trigonometrical arc-functions
<code>atan(a,b)</code>	Returns the principal value of the arc tangent of <i>b/a</i> , expressed in radians (c-function: <code>atan2(a,b)</code> )
<code>sqrt(a)</code>	Square root of <i>a</i>
<code>len(a,b)</code>	Returns hypotenuse: <code>sqrt(a*a+b*b)</code>
<code>len(a,b,c)</code>	<code>sqrt(a*a+b*b+c*c)</code>
<code>pow(a,b)</code>	<i>a</i> to the <i>b</i> power: $a^b$
<code>exp(a)</code>	Exponential function <i>e</i> power <i>a</i> : $e^a$
<code>ln(a), ln2(a), ln10(a)</code>	Log base <i>e</i> / base 2 / base 10 of <i>a</i>
<code>deg(a)</code>	Converts radians into degrees
<code>rad(a)</code>	Converts degrees into radians
<code>return</code>	Return value (optional)
<code>hist(a)</code>	History of <i>a</i> , previous value (if present)
<code>hist(a,n)</code>	Writes <i>n</i> <sup>th</sup> value in history <i>a</i> ( <i>n</i> <= 1000)
<code>avg(a,n)</code>	Moving average of <i>a</i> over <i>n</i> values ( <i>n</i> <1000)
<code>expavg(v,fac)</code>	Exponential moving average of <i>a</i> , with smoothing factor <i>fac</i> to consider the deviation
<code>mean(a)</code>	Arithmetic mean value of <i>a</i>
<code>mean(a, cond)</code>	Arithmetic mean value of <i>a</i> , when <i>cond</i> is true
<code>diff(a)</code>	Difference to the last value: $a(t)-a(t-1)$
<code>diff(a,b)</code>	Differentiate <i>a</i> with respect to <i>b</i> : $da/db$
<code>integral(a,b, cond)</code>	Integral of <i>a(b)db</i> starting with <i>cond==true</i>
<code>change(a)</code>	Detect change of value <i>a</i> , true if value changed
<code>change(a,delta)</code>	Detect change by more than <i>delta</i> : <code>abs(a-a_old)&gt;delta</code>
<code>latch(a,cond)</code>	Record value of <i>a</i> when <i>cond</i> becomes true
<code>Delta2Ev(a, cond1,cond2)</code>	Delta of <i>a</i> between <i>cond1</i> and <i>cond2</i> : $a(cond2) - a(cond1)$
<code>first()</code>	Executed only once (returns true only at first call)
<code>first(cond)</code>	Executed only once, when <i>cond</i> is true (returns true only at first time <i>cond==true</i> )
<code>first(cond, n)</code>	Executed when <i>cond</i> was true for max. <i>n</i> times
<code>mfirst()</code>	Same as <code>first()</code> , but executed each time the maneuver step is started
<code>mfirst(cond)</code>	Same as <code>first(cond)</code> , but executed each time the maneuver step is started
<code>mfirst(cond, n)</code>	Same as <code>first(cond,n)</code> , but counter reseted each time the maneuver step is started
<code>ManJump("str")</code>	Jump to the minimaneuver with label <i>str</i>
<code>ManJump("n")</code>	Jump to minimaneuver with offset <code>+num</code> ( <i>num</i> <0: jump backwards, e.g. -3, <i>num==+0</i>    <i>num ==-0</i> restart current minimaneuver)
<code>ManJump(expr)</code>	Jump to minimaneuver with calculated offset <i>expr</i>

Table 10.22: Realtime Expressions built-in functions

Function	Description
ManJump (sexpr)	Jump to minimaneuver with calculated label <i>sexpr</i>
<a href="#">Log(txt)</a>	Triggers a session log entry with text <i>txt</i> , multiple arguments can be grouped using double quotes
LogWarn(txt)	Triggers a session log entry of type <i>warning</i> with text <i>txt</i> , multiple arguments can be grouped using double quotes
LogErr(txt)	Triggers a session log entry of type <i>error</i> with text <i>txt</i> which causes the simulation to stop. Multiple arguments can be grouped using double quotes
<a href="#">TestLog(kind,txt)</a>	TestLog: Modifies the test status and adds a textual information to the TestManager's result column. Available kinds: info (default), good (green light), bad (red light), warn (yellow light), err (red flash)
<a href="#">pulse(cond)</a>	Instantaneous pulse for one cycle when the condition <i>cond=true</i>
<a href="#">pulse(cond,n)</a>	Instantaneous pulse for <i>n</i> cycles when condition <i>cond=true</i>
LP1(a, RC)	PT1 filter (low pass filter of first order): Filters <i>a</i> with time constant <i>RC</i>
HP1(a, RC)	DT1 filter (high pass filter of first order): Filters <i>a</i> with time constant <i>RC</i>
<a href="#">signal("proc", arg1, arg2 ...)</a>	Executes a procedure <i>proc</i> with the arguments <i>arg1, arg2, etc.</i> defined in a ScriptControl file

### C.2.3 Aliases

Table 10.23: These aliases are accepted by Realtime Expressions

Constant	
e	Euler's number
pi	Circle constant
t	CarMaker quantity Time
s	CarMaker quantity Car.Road.sRoad
cnt	Cycle counter

## C.3 Application Examples

The following examples intend to show the most common applications for RealtimeExpressions.

### C.3.1 Realtime Expressions as Trigger Condition

Realtime Expressions can serve as trigger using logical expressions which are evaluated in every simulation cycle. These expressions can start or finish a maneuver step for the ego vehicle in the Maneuver dialog as well as for traffic objects in the Traffic dialog. Furthermore, the execution of minimaneuver commands or ScratchPad notes in the ScriptControl programming language can be triggered.

Either way, a User Accessible Quantity from the CarMaker Data Dictionary is combined with a logical operator to create a statement which is true or false:

### C.3.2 Maneuver Definition

Realtime Expressions can be used to define the start / end condition in the maneuver description of the ego vehicle and traffic objects. The condition can directly be entered in the prepared parameter field of the dialog.

**Example** The maneuver is started / finished, when the ego car reaches 100kph and has a longitudinal acceleration of less than  $2.5 \text{ m/s}^2$ :  
**Car.v>100/3.6 && Car.ax<=2.5**

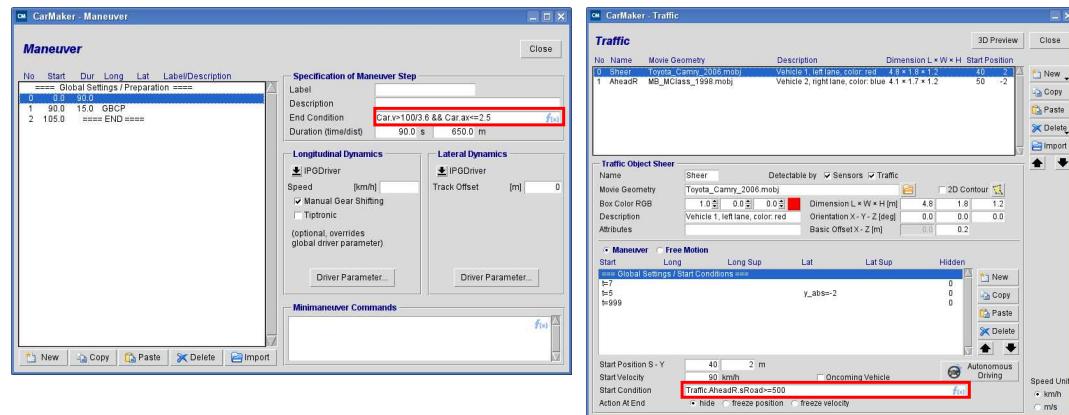


Figure 10.65: Realtime Expression used as maneuver end condition (left) and start condition for a traffic object (right)

Please find further information on the definition of maneuver steps in [section 4.5 'Maneuver' on page 91](#) and about IPGTraffic in [section 4.8 'Traffic' on page 127](#).

### C.3.3 Trigger Minimaneuver Commands

Realtime Expressions can serve as trigger for minimaneuver commands in the Maneuver dialog. The minimaneuver command will be executed, once the condition becomes true for the first time. The trigger condition has to be placed in squared brackets, followed by a blank and the minimaneuver command itself.

**Example** When the brake pedal is activated a message is sent to the session log. The second message is triggered when the vehicle is standing:  
`[change(DM.Brake)] Log "Braking maneuver starts!"`  
`[Car.v<=0.1] Log "Braking finished!"`

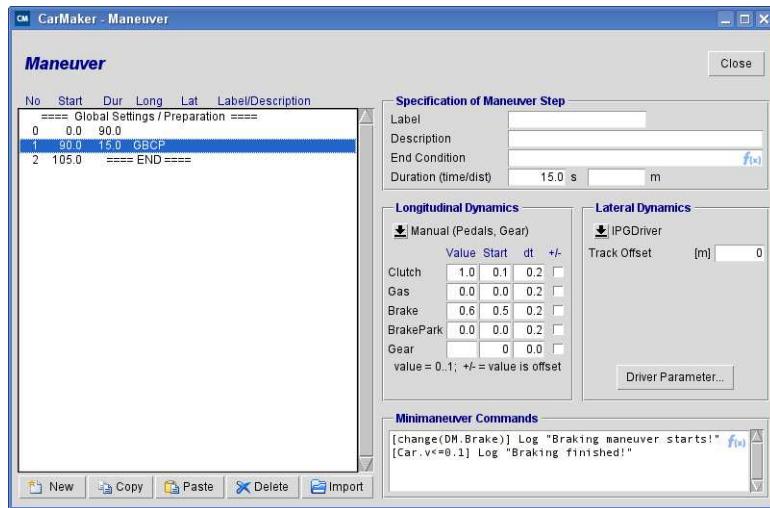


Figure 10.66: Realtime Expressions as trigger for a minimaneuver commands

Please find more information about the Minimaneuver Command Language in [section 4.5.8 'Minimaneuver Command Language' on page 107](#) and a whole command reference in [section B.1 'Command Reference' on page 590](#).

### C.3.4 ScriptControl programming language

Realtime Expressions can trigger *ScratchPad Notes* with a special option to the *QuantAudit* function:

**Example** The command option "-cond" needs to be used:  
`QuantAudit add Car.ax -cond {Car.Roll<-0.01 || Car.Roll>0.01}`

Please find more information on ScratchPad notes and ScriptControl in general, in the Programmer's Guide.

Furthermore, ScriptControl commands or user defined tcl procedures can be called during the simulation using the Realtime Expression command *signal*. Please find further information under section [signal\(\)](#).

### C.3.5 Realtime Expressions in the Minimaneuver Command Language

Realtime Expressions can not only serve to trigger minimaneuver commands. There are several ways to use Realtime Expressions as embedded command in the minimaneuver command language. In that case, the Realtime Expression is indicated by the prefix *Eval*.

**Example**    Realtime Expressions as command in the minimaneuver command language:  
`Eval Qu::var1=integral(PT.Engine.Consump_Abs,Car.Distance,Car.ax<0)`

The following sections present examples on how Realtime Expressions can be used efficiently in the minimaneuver command language.

### C.3.6 Overwrite Quantities

In the minimaneuver command language a *Direct Variable Access* (DVA) command can be imposed which overwrites the value of a User Accessible Quantity from the CarMaker Data Dictionary. As the command itself is quite large, in some cases the same action can be defined much shorter using Realtime Expressions.

**Example**    The following two commands are identical: after 15s simulation time the brake pedal is pushed 50% for a duration of 1s:  
`Eval Time>=15 && Time<=16 ? DM.Brake=0.5  
[Time >=15] DVArw DM.Brake Abs DM 1000 0.5`

### C.3.7 Definition of new Quantities

Sometimes it can be useful to create a new User Accessible Quantity during the maneuver, e.g. to store a certain value of interest at a specific event. The new UAQ will be treated just as any pre-defined UAQ, it will be available to other applications such as IPGControl or for Direct Variable Access.

The declaration of a new quantity is indicated by the prefix "Qu::" and the name of the quantity:

**Example**    `Eval Qu::var1=10`

Now a new quantity named *var1* of type double was created that has the value 10.

Each quantity can be combined with any Realtime Expressions function:

**Example**    `Eval change(DM.Brake) ? Qu::v_crit = Car.v`

In this example a new quantity called *v\_crit* is created which saves the current velocity of the vehicle at the moment the brake pedal is pushed.

In the Realtime Expressions syntax there is no need to declare a new quantity before usage - both can be done in one step.



### C.3.8 Definition of new Variables

Instead of User Accessible Quantities you can create temporary variables, too, which only exist during the simulation within the same minimaneuver command they are created in. Temporary variables cannot be accessed by any other application than CarMaker (e.g. not visible in IPGControl). The syntax and usage of temporary variables is quite similar to the UAQ definition, it is just the prefix "Qu::" which is skipped:

**Example**    `Eval temp=20`

Please note: a point in the variable name (e.g. my.var) is not permitted.



## C.3.9 Syntax Examples

### Round Brackets

Round brackets can be used to group statements which are then executed at first.

**Example** Calculate the mean slip angle at the front axle:

```
Eval Qu::SlipAngle_F=0.5*(Car.SlipAngleFR+Car.SlipAngleFL)
```

### Conditional Expression

If-then-else command:

**Example** If a given lateral acceleration is exceeded, activate the handbrake else do not activate the handbrake, whereas the "else"-part is optional:

```
Eval Car.ay>=3.0 ? DM.Handbrake=1.0 : DM.Handbrake=0
```

### Sequence

List more than one command in one line:

**Example** If the vehicle velocity exceeds 60kph, activate the clutch and brake pedal at the same time:

```
Eval Car.v>=60/3.6 ? (DM.Brake=0.7; DM.Clutch=1.0)
```

### ceil(a)

Round up function:

**Example** Least integral value greater than or equal to the argument:

```
Eval MyVar=ceil(4.5) -> returns MyVar=5
```

```
Eval MyVar=ceil(4.0) -> returns MyVar=4
```

```
Eval MyVar=ceil(-4.5)-> returns MyVar=-4
```

### floor(a)

Round down function:

**Example** Greatest integral value less than or equal to the argument:

```
Eval MyVar=floor(4.5) -> returns MyVar=4
```

```
Eval MyVar=floor(4.0) -> returns MyVar=4
```

```
Eval MyVar=floor(-4.5)-> returns MyVar=-5
```

### trunc(a)

Cuts the positions after decimal point, keeping the sign:

**Example** Round argument to the nearest integer not larger in abs value:

```
Eval MyVar=trunc(-4.5)-> returns MyVar=-4
```

**hist(a,n)** Create a signal history for a filling the n<sup>th</sup> element in the list:

**Example**

```
Eval prev=hist(cnt,3)
cycle no (cnt)=> prev
0: => 0
1: => 0
2: => 0
3: => 0
4: => 1
5: => 2
6: => 3
7: => 4
8: => 5
9: => 6
10: => 7
```

**integral(a,b, cond)** Build the integral of a function over a quantity (first argument) with start condition (second argument):

**Example** Calculate the travelled distance after simulation time exceeded 5s:  
`Eval MyDistance=integral(Car.v, Time, Time>5)`

**change(a)** Observe state of a quantity:

**Example** Open the clutch whenever the brake pedal is activated:  
`Eval change(DM.Brake) ? DM.Clutch=1.0`

**Example** Open the clutch only at full braking:  
`Eval change(DM.Brake,0.7) ? DM.Clutch=1.0`

**latch(a,cond)** This command records the value of a quantity if the condition is true:

**Example** Create a new quantity which records the value of the steering wheel angle if the lateral acceleration exceeds 3.0 m/s<sup>2</sup>:  
`Eval Steer_crit=latch(DM.SteerAng, Car.ay>=3.0)`

**Delta2Ev(a, cond1,cond2)** This command computes the difference of a value between two events:

**Example** Calculate the braking distance: Difference of vehicle distance between the first actuation of the brake pedal and the standstill of the car:  
`Eval s_stop = Delta2Ev(Car.Road.sRoad, change(DM.Brake), Car.v<=0.1)`

**first()** Executes the argument only once during the maneuver:

**Example** Initiates a new quantity by assigning a default value at the start of the maneuver:  
`Eval first() ? Qu::myvar=0`

**ManJump("str")** Minimaneuver jump to minimaneuver with label str:

**Example** Jump to minimaneuver with label "Accel", when velocity is lower than 3m/s:  
`Eval Car.v<=3.0 ? ManJump("Accel")`

**ManJump("n")** Relative maneuver jump with:

Table 10.24: Maneuver jump options

argument	effect
$n < 0$	jump backwards, e.g. -2
$n > 0$	jump forwards, e.g. +2
$n == +0 \text{    } n == -0$	restart same minimaneuver
$n >>$	offset larger than highest maneuver number: jump to last minimaneuver
$n <<$	offset smaller than first maneuver number: jump to first minimaneuver
$n = \text{num}$	absolute jump to maneuver <i>num</i>

**Example** Jump two maneuver steps further from current maneuver:

```
Eval Car.v<=3.0 ? ManJump("+2")
```

**ManJump (expr)** The maneuver step can also be calculated:

**Example** Jump from current maneuver two steps back:

```
Eval ManJump(DM.ManNo-2)
```



Please note that the maneuver jump is executed not before the start of the next simulation cycle after the condition was true. This means, all Realtime Expressions that follow are still executed. The maneuver jump can also be overruled by a following command.

**Log(txt)** Writes to the session log. Use double quotes to group strings. Different formats are possible:

**Example** Writes text to the session log:

```
Eval first() ? Log("TestRun started!")
```

**Example** Writes a warning to the session log which is marked in yellow and can be detected by scripts:

```
Eval first(Car.ay>=3.0) ? LogWarn("Critical lateral acceleration exceeded!")
```

**Example** Writes an error message to the session log which causes the simulation to stop:

```
Eval first(Driver.Lat.dy>=2.0) ? LogErr("Driver leaves road!")
```

```
SIMULATE Examples/VehicleDynamics/SteadyCircle_100m
TestRun started!
Time 40.716
WARNING Critical lateral acceleration exceeded!
Time 64.187
ERROR Driver leaves road!
SIM_ABORT Examples/VehicleDynamics/SteadyCircle_100m 64.188s 877.023m
```

Figure 10.67: Session log output created by Realtime Expressions



Hint: Use the Log() function in combination with the first() function, otherwise the log messages are created each simulation cycle / each time the condition becomes true.

The Log() command can also be used to print the value of a quantity. The syntax is similar to the c-code language: The quantity must not be included in the double quotes which contain the text output, but separated by comma instead. Different formats can be used - without any format definition the default settings (empty string followed by default number format) are used.

**Example** Write current value of steering wheel angle to the Session Log when lateral acceleration limit is reached:

```
Eval first(Car.ay>=3.0) ? LogWarn("Critical lateral acceleration exceeded at steering wheel angle:", DM.Steer.Ang)
```

SIMULATE Examples/VehicleDynamics/Road\_Hockenheim\_R8  
Time 25.828  
**WARNING** Critical lateral acceleration exceeded at steering wheel angle: 0.418898  
Time 79.127  
SIM\_END Examples/VehicleDynamics/Road\_Hockenheim\_R8 79.119s 2412.71m

Figure 10.68: Value output of a quantity in the Session Log

**Example** Write current value of steering wheel angle to the Session Log with three decimal places:

```
Eval first(Car.ay>=3.0) ? LogWarn("Critical lateral acceleration exceeded at steering wheel angle: %.3f", DM.Steer.Ang)
```

SIMULATE Examples/VehicleDynamics/Road\_Hockenheim\_R8  
Time 25.828  
**WARNING** Critical lateral acceleration exceeded at steering wheel angle: 0.419 rad  
Time 65.950  
SIM\_END Examples/VehicleDynamics/Road\_Hockenheim\_R8 65.942s 2010.16m

Figure 10.69: Value output of a quantity in the Session Log with user defined formatting

**TestLog(kind,txt)** Triggers a balloon information in the TestManager's result column and modifies the result status. Possible kinds:

Table 10.25: Available status kinds for TestLog command

kind	status / bulb color
info	default, light not changed
good	status: good, green light
bad	status: bad, red light
warn	status: warning, yellow light
err	status: error, red flash

**Example** The following command turns the result status to red (bad), if the roll angle exceeded 3 degrees:

```
Eval mfirst(Car.Roll>=3*pi/180) ? TestLog("bad","Roll angle limit exceeded")
```

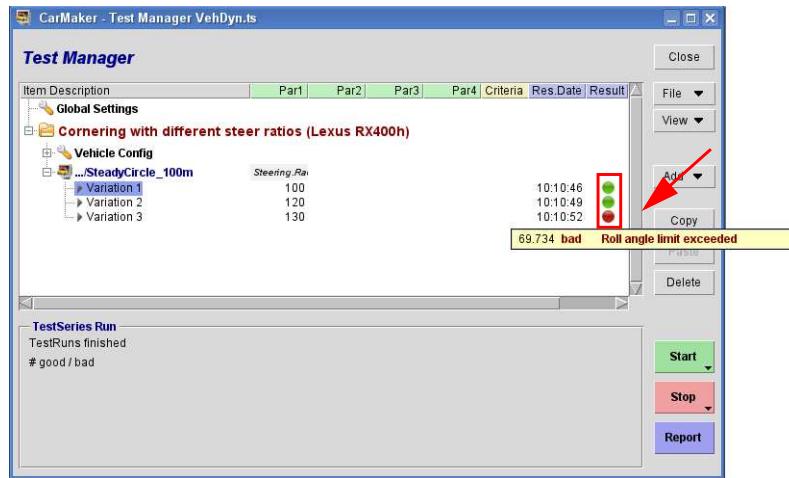


Figure 10.70: Modification of the result status in the TestManager

- pulse(cond)** Creates an instantaneous pulse of 1.0 for one cycle when the condition becomes true. This command can serve as trigger condition:

**Example** Save the vehicle speed in that very cycle the brake pedal is activated by 10%:  
`Eval pulse(DM.Brake>0.1) ? MaxSpeed=Car.v`

- pulse(cond,n)** Provides an instantaneous pulse for a specified number of cycles when the condition becomes true.

**Example** Overwrite the brake pedal for 10 simulation cycles when a certain vehicle speed is reached:  
`Eval pulse(Car.v>50/3.6, 10) ? DM.Brake=0.7`

- signal()** Procedures defined in ScriptControl can be called during the simulation using the Realtime Expression *signal*. This functionality is designed for triggering single events during the simulation that are not time critical, as there may be a delay between calling the procedures with *signal* and the execution of the procedure. Therefore, this functionality is best used for executing postprocessing or locking functions and sending commands and information to other tools and scripts.



Please note, that the Realtime Expression *signal* can be executed only a limited number of times during 1 second. That is why you should avoid calling it every simulation cycle.

The ScriptControl procedures that should be used for a TestRun are defined in one script file and stored inside the folder Data/Script of the project directory. This file than needs to be selected as a *ScriptControl File* in the *Global Settings* of the maneuver. Please take care to define the ScriptControl procedures in this file using the command *SigHandler* (instead of *proc*) as shown in Figure 10.71.

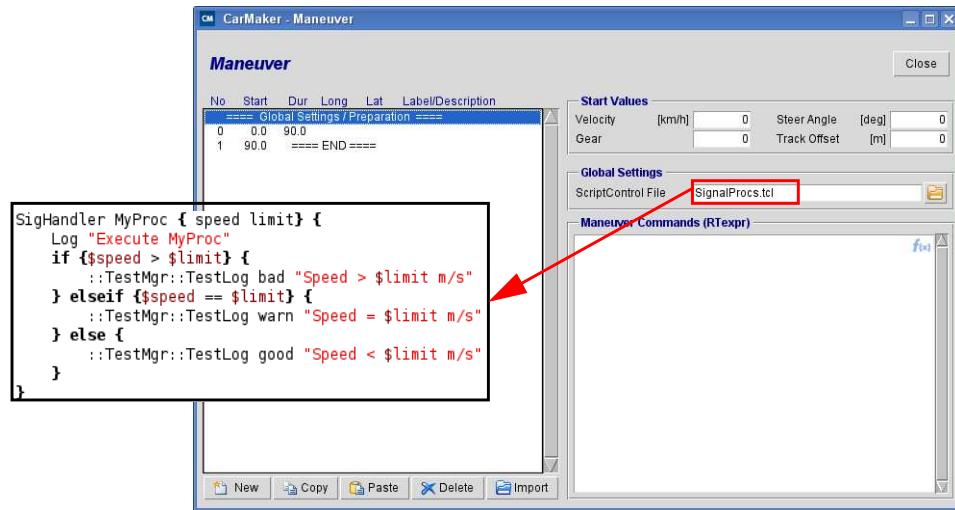


Figure 10.71: Defining ScriptControl procedures for Realtime Expression signal

To execute this example procedure of [Figure 10.71](#), the Realtime Expression *signal* is used as follows.

**Example**      The first argument of *signal* is the procedure name, the other arguments correspond to the procedures arguments:  
`Eval first() ? signal("MyProc", Car.v, 10)`

## Appendix D

# Model Check Quantity List

This appendix contains the Quantity description for the result files generated for the Model Check diagrams. They can be used for postprocessing purposes just like the simulation \*.erg result files. Moreover, a table for the tire result file name description is included at the end.

### Aerodynamics

Table 10.26: Model Check, Aerodynamics Signal List

Signal Name	Unit	Description
<dir> := x, y, z (Axis)		
tau	deg	Angle of wind approach direction
cDragA	m <sup>2</sup>	Vehicle reference area coefficient
cDragF	-	Total drag force coefficient
cLiftF	-	Total lift force coefficient
cLiftF_front	-	Front lift force coefficient
cLiftF_rear	-	Rear lift force coefficient
cSideF	-	Total side force coefficient
cSideF_front	-	Front side force coefficient
cSideF_rear	-	Rear side force coefficient
cRollTrq	-	Roll coefficient
cPitchTrq	-	Pitch coefficient
cYawTrq	-	Yaw coefficient
Frc_1.<dir>	N	Aerodynamic force in <dir>-direction in point of attack A
Trq_1.<dir>	Nm	Aerodynamic torque around <dir>-Axis in point of attack A
vres_1.<dir>	m/s	Relative velocity wind to vehicle in <dir>-direction in point of attack A

## Engine

Table 10.27: Model Check, Engine Signal List

Signal Name	Unit	Description
<dir> := F, B (Forward, Backward) <i> := 1, 2, 3, 4.. (Gear No)		
nRot	rpm	Engine rotational speed
Gas	-	Accelerator pedal
Torque	Nm	Engine torque
Power	kW	Engine power
<dir>Gear.<i>.vVehicle	km/h	Vehicle velocity
<dir>Gear.<i>.Frc_drive	N	Tractive force, climbing resistance
<dir>Gear.<i>.kappa	1/m	Max Tractive Force / Max Engine Torque @ Gas =1.0
<dir>Gear.<i>.Pwr_drive	kW	Tractive power of <i> gear

## Tire

Table 10.28: Model Check, Tire Signal List

Signal Name	Unit	Description
<dir> := x, y, z (Axis)		
vBelt	m/s	Velocity tire belt
muRoad	-	Coefficient of friction
Inclinangle	deg	Inclination angle
Alpha	deg	Side slip angle
Slip	%	Longitudinal slip
rBelt_eff	m	Effective tire belt radius
TurnSlip	1/m	Turn slip
Rim_rotv	rad/s	Rim rotation velocity
Frc_W2Frc_z	-	Equivalent Force (Frc_W_x & Frc_W_y) / Vertical Load on Tire (Frc_W_z)
P_v0_W.<dir>	m/s	Vehicle Speed for Model Check
Frc_W.<dir>	N	Effective force
Trq_W.<dir>	Nm	Effective torque

## Brake

Table 10.29: Model Check, Brake Signal List

Signal Name	Unit	Description
<pos> := FL, FR, RL, RR (Front Left, Front Right, Rear Left, Rear Right) <no> := 2 (If more than 1 cylinder)		

Table 10.29: Model Check, Brake Signal List

Signal Name	Unit	Description
Brake.Pedal	-	Brake pedal position (0-1)
Brake.Park	-	Park Brake position (0-1)
Brake.pWB_<pos><no>	bar	Pressure
Brake.pGrad_<pos><no>	bar/s	Pressure gradient
Brake.Trq_WB_<pos>	Nm	Torque at Wheel
Brake.Trq_<pos><no>_ext	Nm	External torque at Wheel
Brake.pGrad_<pos>	bar/s	Brake pressure speed
Brake.pGrad_<pos>2	bar/s	Brake pressure speed trailer

## IPGDriver

Table 10.30: Model Check, IPGDriver Signal List

Signal Name	Unit	Description
<pos>:=l, c, r (left, center, right) <dir> := x, y (Axis)		
Road.<pos>.<dir>	m	Road coordinates
SDC.Track.<pos>.<dir>	m	Static-desired-course-track coordinates
SDC.<dir>	m	Static-desired-course coordinates
SDC.s	m	S-course
SDC.v	m/s	Velocity
SDC.ax	m/ s^2	Lateral acceleration
SDC/ay	m/ s^2	Longitudinal acceleration
SDC.dz	-	Change in SDC.z / SDC.z
SDC.ddz	-	Change in SDC.dz / SDC.dz
SDC.Weight	-	Weight of Static Desired Course spline.
SDC.Curv	1/m	Road curvature
SDC.Track.<pos>Width	m	Static-desired-course-track width
SDC.Track.lWidth	m	SDC Track Width left
SDC.Track.rWidth	m	SDC Track Width right
SDC.Track.Width	m	SDC Total Track Width
SDC.Track.<pos>.<dir>	m	SDC position from Road Origin
Road.s	m	S-road
Road.Slope	-	Road slope
Road.Grad	-	Road grad
Road.Side	m	Lateral deviation of Car from centre line
Road.Width	m	Road total width
Road.l.Width	m	Road width left
Road.r.Width	m	Road width right

## Suspensions Force ElementsOutput signals of ModelCheck

Table 10.31: Model Check, Suspension Force Elements Signal List

Signal Name	Unit	Description
<pos> := FL, FR, RL, RR (Front Left, Front Right, Rear Left, Rear Right) <dir> := x, y, z (Axis)		
AxleFrc<pos>	N	Force on <pos> in z direction
TrackCircleDiam	m	Turning Track Diameter of Outer Wheel
Susp<pos>.q0	mm	Translational movement of <pos> wheel carrier with respect to q0
Susp<pos>.q0p	m/s	Velocity of <pos> damper with respect to q0
Susp<pos>.GenFrc0	N	Generalised Suspension Force on <pos> in Global Frame
Susp<pos>.GenFrc1	N	Generalised Effective Suspension Force on <pos> in Vehicle Frame
Susp<pos>.SecSpringFrc0	N	Generalised Suspension Force of Secondary Spring on <pos> in Global Frame
Susp<pos>.Frc2CExt_1.<dir>	N	External forces to the wheel carrier on <pos>, defined in vehicle frame in <dir>
Susp<pos>.Trq2CExt_1.<dir>	Nm	External torque to the wheel carrier on <pos>, defined in vehicle frame in <dir>
Susp<pos>.C_1.<dir>	m	<pos> Wheel carrier co-ordinates in vehicle frame in <dir>
Susp<pos>.P_1.<dir>	m	<pos> Tire contact point co-ordinates in vehicle frame in <dir>
Susp<pos>.Mount2P_1.<dir>	mm	<pos> Strut mount co-ordinates with respect to Tire contact point in vehicle frame in <dir>
Susp<pos>.CastorAng	deg	Castor Angle of <pos>
Susp<pos>.KingpinAng	deg	Kingpin Angle of <pos>
Susp<pos>.CastorOffset	mm	Caster Offset of <pos>
Susp<pos>.KingpinOffset	mm	Kingpin Offset of <pos>
Susp<pos>.t<dir>	mm	Translational movement of <pos> wheel carrier in <dir>
Susp<pos>.t<dir>_kin	mm	Translational movement of <pos> wheel carrier in <dir> by Kinematics
Susp<pos>.t<dir>_com	mm	Translational movement of <pos> wheel carrier in <dir> by Compliance
Susp<pos>.t<dir>_ext	mm	Translational movement of <pos> wheel carrier in <dir> external offset
Susp<pos>.r<dir>	deg	Rotational movement of <pos> wheel carrier around <dir>
Susp<pos>.r<dir>_kin	deg	Rotational movement of <pos> wheel carrier around <dir> by Kinematics

Table 10.31: Model Check, Suspension Force Elements Signal List

Signal Name	Unit	Description
Susp<pos>.r<dir>_com	deg	Rotational movement of <pos> wheel carrier around <dir> by Compliance
Susp<pos>.r<dir>_ext	deg	Rotational movement of <pos> wheel carrier around <dir> external offset
Susp<pos>.SecSpringFrc0	F	Secondary spring force at <pos>
Damp<pos>.l	mm	Length of <pos> damper
Damp<pos>.v	m/s	Velocity of <pos> damper
Damp<pos>.dldz	mm/ mm	Damper length rate, delta of spring length to delta of translational movement in z direction
Damp<pos>.Frc	N	Damper Force at <pos>
Damp<pos>.Frc_tot	N	Total damper force at <pos>
Damp<pos>.Frc_ext	N	External damper force at <pos>
Spring<pos>.l	mm	Length of <pos> spring
Spring<pos>.dldz	mm/ mm	Spring length rate, delta of spring length to delta of translational movement in z direction
Spring<pos>.Frc	N	Spring force at <pos>
Spring<pos>.Frc_tot	N	Total spring force at <pos>
Spring<pos>.Frc_ext	N	External spring force at <pos>
Buffer<pos>.l	mm	Buffer length at <pos>
Buffer<pos>.dldz	mm/ mm	Buffer length rate, delta of spring length to delta of translational movement in z direction
Buffer<pos>.Frc	N	Buffer force at <pos>
Buffer<pos>.Frc_ext	N	External Buffer force at <pos>
Buffer<pos>.Frc_tot	N	Total Buffer force at <pos>
Stabi<pos>.l	mm	Stabilizer length at <pos>
Stabi<pos>.dldz	mm/ mm	Stabilizer length rate, delta of Stabilizer length to delta of translational movement in z direction
Stabi<pos>.Frc	N	Stabilizer force at <pos>
Stabi<pos>.Frc_ext	N	External Stabilizer force at <pos>
Stabi<pos>.Frc_tot	N	Total Stabilizer force at <pos>

## Kinematic and Compliance

Signal Name	Unit	Description
<pos> := FL, FR, RL, RR (Front Left, Front Right, Rear Left, Rear Right)		
<posA>:= F, R (Front, Rear)		
<dir> := x, y, z (Axis)		

Signal Name	Unit	Description
Toe<pos>	min	Toe Angle of left/right wheel carrier Positive sign means toe-in, negative sign means toe-out.
SteerAngle<pos>	min	Steer angle (ISO 8855, 4.1.5.1)
Camber<pos>	deg	Camber angle CamberAngle wheel. Positive sign means the distance on the upper side of the wheel is longer then on the bottom side.
SpinAngle<pos>	deg	Spin Angle at <pos>
InclinAngle<pos>	deg	Inclination Angle at <pos>
Susp<pos>.q0	mm	Movement of Suspension <pos> with respect to the generalized coordinate q0
Susp<pos>.q0p	m/s	Velocity of Suspension <pos> with respect to the generalized coordinate q0
Susp<pos>.q1	mm	Movement of Suspension <pos> with respect to the generalized coordinate q1
Susp<pos>.q2	mm	Movement of Suspension <pos> with respect to the generalized coordinate q2
AxleForce<pos>	N	Force on Axle at <pos> in z direction
Susp<pos>.t<dir>	mm	Translational movement of <pos> wheel carrier in <dir> direction
Susp<pos>.r<dir>	deg	Rotational movement of <pos> wheel carrier around <dir> direction
Steer.txRack	mm	Translational Movement of Steering Rack in x direction
Steer.Susp<pos>.drz_dqSteer	deg/mm	Delta of SuspFL.rz and SuspFR.rz / Delta of Steering Rack position
Steer.Susp<pos>.drz_dSteerAng	-	Delta of SuspFL.rz and SuspFR.rz / Delta of Steering Wheel Angle
Steer.i_qSteer2rz	deg/mm	Steering Wheel Angle / Rack displacement
Steer.i_StAng2rz	-	Steering Wheel Angle / Tire Angle in Z
Steer.dqSteer_dSteerAng	deg/mm	Delta of Steering Rack position / Delta of Steering Wheel Angle
Steer.drz	deg	Delta of SuspFL.rz and SuspFR.rz
Steer.q	mm	Steering Rack position
Steer.SteerAng	deg	Steering Wheel Angle
Susp<pos>.t<dir>	mm	Translation of carrier reference point at <pos>
Susp<pos>.t<dir>_kin	mm	Translation of carrier reference point at <pos> by Kinematics
Susp<pos>.t<dir>_com	mm	Translation of carrier reference point at <pos> by compliance
Susp<pos>.t<dir>_ext	mm	Translation of carrier reference point at <pos> external
Susp<pos>.r<dir>	deg	Rotational movement of <pos> wheel carrier around <dir>

Signal Name	Unit	Description
Susp<pos>.r<dir>_kin	deg	Rotational movement of <pos> wheel carrier around <dir> by Kinematics
Susp<pos>.r<dir>_com	deg	Rotational movement of <pos> wheel carrier around <dir> by Compliance
Susp<pos>.r<dir>_ext	deg	Rotational movement of <pos> wheel carrier around <dir> external offset
Susp<pos>.Frc2CExt_1.<dir>	N	Force at <pos> Wheel Carrier in <dir> direction
Susp<pos>.Trq2CExt_1.<dir>	N	Torque at <pos> Wheel Carrier in <dir> direction
Susp<pos>.C_1.<dir>	mm	Displacement in <dir> of wheel carrier C at Suspension <pos>
Susp<pos>.P_1.<dir>	mm	Displacement in <dir> of tire contact point P at Suspension <pos>
Susp<pos>.Mount2P_1.<dir>	mm	Delta between Wheel Mount and Contact Point at <pos> in <dir>
Susp<pos>.CastorAng	deg	Castor Angle of <pos>
Susp<pos>.KingpinAng	deg	Kingpin Angle of <pos>
Susp<pos>.CastorOffset	mm	Caster Offset of <pos>
Susp<pos>.KingpinOffset	mm	Kingpin Offset of <pos>
KinRollCenter_<posA>.<dir>	m	Kinematic Roll Center of Axle <pos> in direction <dir>
Susp<posA>.TrackWidth	m	Track Width of <posA> Axle
Susp<posA>.WheelBase	m	Wheel Base of <posA> Axle
TrackCircleDiam	m	Turning Track Diameter of Outer Wheel

## Kinematic and Compliance SPMM

Table 10.32: Model Check KnC SPMM Signal List

Signal Name	Unit	Description
<pos> := lf, rf, lr, rr (Left Front, Right Front, Left Rear, Right Rear)		
<posA> := f, r (Front, Rear)		
<dir> := x, y, z (Axis)		
tbounce	mm	Body Bounce Displacement - Vertical displacement of the testrig table centre, in ground axes (positive is rebound)
troll	deg	Body Roll Angle - Angle between the table and ground Y-axis, with respect to rotation about the table X-axis (positive is right side down)
tpitch	deg	Body Pitch Angle - Angle between testrig table and ground X-axis, with respect to rotation about the Y-axis
srwtc	-	Static Roll Weight Transfer Coefficient (SRWTC)
tmx	Nm	Axle Roll Moment
swa	deg	Steering Wheel Angle
swt	Nm	Steer Wheel Torque

Table 10.32: Model Check KnC SPMM Signal List

Signal Name	Unit	Description
steeracker	deg	Steer (Ackermann)
steerratio	-	Steering Ratio
ctf<dir>	N	Total Force at all four Tire contact patch in <dir>
ctm<dir>	Nm	Total Moment at all four Tire contact patch in <dir>
<pos>steerangle	deg	Steer angle at tire
<pos>steer	deg	Wheel Toe-in, rotational movement around z axis
<pos>cam	deg	Wheel Camber, rotational movement around x axis
<pos>ry	deg	Wheel Castor, rotational movement around y axis
<pos>spin	deg	Wheel Spin, rotational movement around z axis
<pos>wbz	mm	Wheel to body displacement on z-Axis
<pos>tc	mm	Tire compression
<pos>f<dir>	N	Force at wheel for <pos> in <dir>
<pos>m<dir>	Nm	Moment at Wheel for <pos> in <dir>
<pos>wcd<dir>	mm	Wheel center displacement on <dir>-Axis
<pos>wc<dir>	mm	Wheel Center on <dir>-Axis
<pos>cp<dir>	mm	<Pos> Tire contact coordinates in <dir>
<pos>cpd<dir>	mm	Change in <Pos> Tire contact coordinates in <dir>
<pos>_krch	mm	Kinematic roll center height
<pos>vsal	mm	Virtual Swing Arm Length of <pos>
<pos>vsaa	deg	Virtual Swing Arm Angle of <pos>
<pos>scrubradius	mm	Scrub Radius at <pos>
<pos>cas	deg	Castor Angle at <pos>
<pos>trail	mm	Castor Trail at <pos>
<pos>kpangle	deg	Kingpin Angle for <pos>
<pos>kpoffset	mm	Kingpin Offset for <pos>
<pos>kpgrd<dir>	mm	Kingpin Gradient for <pos> in <dir>
<pos>rch	mm	Rollcentre height for <pos>
<pos>_antidive_angle	deg	Antidive Angle for <pos>
<pos>_antisquat_angle	deg	Antisquat Angle for <pos>
<posA>track	mm	Track Change for <posA>
<posA>steerangle	deg	Axle steer angle for <posA>
<posA>sroll	deg	Suspension roll angle for <posA>
<posA>mx	Nm	Axle roll moment for <posA>
<posA>_k_rcy	mm	Kinematic Roll centre coordinate in Y for <posA>
<posA>_k_rcz	mm	Kinematic Roll centre coordinate in Z for <posA>
<posA>_rcy	mm	Roll centre coordinate in Y for <posA>
<posA>_rcz	mm	Roll centre coordinate in Z for <posA>
lwbase	mm	Wheelbase change left side
rwbase	mm	Wheelbase change right side

## Tire result file name description

File Name	Description
* := index <param>:= g, s, ts (inclination angle, longitudinal slip, turn slip)	
Tire_Slp00*_Fz_m000.erg	Not used
Tire_Alpha00*_Fz_m000.erg	Not used
Tire_Fz00*_Alpha_<param>00*_m000.erg	Side force vs. slip angle with variations in <param>
Tire_Fz00*_slp_m00*.erg	Long. force vs. long. slip with variations of road friction coeff. mue
Tire_Fz00*_Alpha_m00*.erg	Side force vs. slip angle with variations of road friction coeff. mue
Tire_Fz00*_slp_a00*_m000.erg	Side force and self aligning torque vs. long. slip with variations of the slip angle