IPG AUTOMOTIVE

Reference Manual Version 7.1.1

CarMaker®

SOLUTIONS FOR VIRTUAL TEST DRIVING

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 the IPG Automotive Group. IPG Automotive Group 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 Group.

© 1999 - 2018 by IPG Automotive Group – 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 the IPG Automotive Group.

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

All other product names are trademarks of their respective companies.

IPG Automotive Group means any corporate body of which IPG Automotive GmbH has a majority stake.

Contents

1 Introduction 17

1.1 General Remarks 17

1.2 CarMaker Axis Systems 18

1.3 CarMaker Geometry Input 20

1.3.1 Conventions defining CarMaker Objects 20

1.3.2 Guidelines defining CarMaker Objects 20

1.4 CarMaker Naming Conventions 22

1.4.1 CarMaker Subsystems 22

1.4.2 Classification of Quantities 23

1.4.3 Meaning of Abbreviations 24

2 CarMaker Parameter Files 26

2.1 Infofile parameter format 26

2.2 Common parameter entries 26

2.2.1 File identification 26

2.2.2 Model kind 27

2.2.3 Inclusion of Infofiles 28

2.3 Selected TestRun parameters 28

2.3.1 Vehicle and trailer loads 28

2.3.2 Random number generator 29

3 CarMaker Configuration 30

3.1 SimParameters 31

3.1.1 General application parameters 31

3.1.2 Cycle timing 31
3.1.3 Cycle timing (CarMaker/HIL) 32
3.1.4 Monitoring Realtime Conditions (CarMaker/HIL) 32
3.1.5 Data Storage 33
3.1.6 Session / TestRun commands 34
3.1.7 Vehicle Model 34
3.1.8 Tire Model MF-Tyre/MF-Swift 35
3.1.9 Maneuver Control 36
3.1.10 Start of TestRun 37
3.1.11 Vehicle Reload 38
3.1.12 End of test run 39
3.1.13 Pylon detection 40
3.1.14 QuantAudit 40
3.1.15 Scratchpad 40
3.1.16 Geographic coordinate system 41
3.1.17 ADTF Interface 41
3.1.18 CRUISE Interface 41
3.1.19 GT-SUITE Interface 42
3.1.20 Video Data Stream Client 43
3.1.21 ADAS RP 44
3.1.22 Functional Mock-up Interface (FMI) 44
3.1.23 CarMaker for Simulink 45
3.1.24 Generic Plug-in Model 46
3.1.25 GPU Sensors 46
3.2 ECUParameters 47
3.3 OutputQuantities 47
3.4 Archive 48

4 Environment 49
4.1 General Parameters 49
4.2 Environment Model "Generic" 52
4.2.1 Model description 52
4.2.2 Environment Model "Generic" Parameters 53
4.3 Material Parameters 54
4.4 User Accessible Quantities 55

5 Driving Maneuvers 57
5.1 Overview 57
5.2 Longitudinal Dynamics Maneuvers 59

5.2.1 IPGDriver 59
5.2.2 Stop Vehicle 59
5.2.3 Drive backwards 59
5.2.4 Manual 59
5.2.5 Speed Control 60
5.2.6 Speed Profile 60
5.3 Lateral Dynamics Maneuvers 61
5.3.1 IPGDriver 61
5.3.2 Steer Step 61
5.3.3 Sine Steering 62
5.3.4 Sine Sweep 62
5.3.5 Simple Steering Control (Follow Course) 62
5.4 Special Maneuvers 62
5.4.1 The last Mini Maneuver (Duration 0 Seconds) 62
5.5 Vehicle Operator 63
5.6 User Accessible Quantities 66

6 VehicleControl 67
6.1 Introduction 67
6.2 VehicleControl Interface 67
6.3 General Parameters 68
6.4 ACC Controller 69
6.4.1 ACC Controller Model 69
6.4.2 ACC Controller & VehicleControl *AccelCtrl* *69*
6.4.3 Model Parametrization 70
6.4.4 User Accessible Quantities 72
6.5 Generic Longitudinal Control 73
6.5.1 Autonomous Emergency Braking (AEB) 73
6.5.2 Forward Collision Warning (FCW) 73
6.5.3 Model Parametrization 74
6.5.4 User Accessible Quantities 75
6.6 Generic Lateral Control 76
6.6.1 Lane Keeping Assist System (LKAS) 76
6.6.2 Lane Departure Warning (LDW) 76
6.6.3 Model Parametrization 77
6.6.4 User Accessible Quantities 78

7 Vehicle Body 79
7.1 Overview 79

7.1.1 Configuration of the vehicle model 80
7.1.2 Finding the equilibrium state 80
7.1.3 Vehicle Interface 81
7.1.4 Interaction with other modules 81
7.2 General Parameters 82
7.3 Mass Geometry 84
7.3.1 Overview Mass Geometry 84
7.3.2 Parameters 84
7.3.3 Vehicle with Twin Tires 86
7.4 User Accessible Quantities for Vehicle Body 87

8 Vehicle Body Modelling 88
8.1 Equations 88
8.2 Model description 89
8.2.1 Architecture of forces 89
8.2.2 Chain of calculation 90
8.2.3 Description of the calculation parts 91

9 Vehicle Model ’Flexible’ 101
9.1 Overview 101
9.2 Parameters – Vehicle Parameter Set 102
9.3 Parameters – Test Run Parameter Set 106
9.4 Examples 106
9.5 User Accessible Quantities 107

10 Elastically mounted bodies 108
10.1 Engine 108
10.1.1 Parameters 109
10.1.2 User Accessible Quantities 111

11 Suspension Force Elements 112
11.1 External Suspension Forces 113
11.1.1 General Parameters 114
11.1.2 Model "SuspComponent" 114
11.2 Springs 115
11.2.1 Coil Spring 115
11.2.2 Secondary Spring 117
11.3 Dampers 118
11.3.1 Damper 118

11.3.2 Damper with Top Mount 121
11.4 Buffers / Bumpers 123
11.5 Suspension Roll Stabilizer / Anti-Roll Bar 128
11.6 Wheel Bearing Friction 131

12 Suspension Kinematics and Compliance 134
12.1 Overview 134
12.1.1 Describing Kinematics with Generalized Coordinates 139
12.1.2 Brief Introduction to the Measurement Procedure of K&C parameters 142
12.2 Kinematics and Compliance 145
12.3 Kinematics Models 147
12.3.1 “Linear 1 DOF”, “Linear 2 DOF” and “Linear 3 DOF” 147
12.3.2 “MapNL” 150
12.4 Compliance Models 158
12.4.1 “Linear Frame Fr1” and “Linear Frame Fr2” 158
12.4.2 “Coeff1DFr1” and “Coeff1DFr2” 160
12.4.3 “Displace1DFr1” and “Displace1DFr2” 162
12.4.4 “Displace2DFr1” and “Displace2DFr2” 164
12.4.5 “SetZero” 167
12.4.6 Example: Superposition of Compliance Models 168
12.5 Suspension Interface 169
12.5.1 Overview 169
12.5.2 Kinematic characteristic parameters 170
12.6 MBS Suspension 173
12.6.1 Overview 173
12.6.2 General Parameters 175
12.6.3 McPherson suspension 180
12.6.4 McPherson extended suspension 186
12.6.5 Fourlink suspension 190
12.6.6 Fourlink extended suspension 198
12.6.7 Twistbeam suspension 201
12.6.8 Twistbeam extended suspension 206
12.6.9 Double wishbone suspension 207
12.6.10 Double wishbone extended suspension 215
12.7 Additional External Movement 219

13 Aerodynamics 221
13.1 Overview 221
13.2 General Parameters 223
13.3 Models 224
13.3.1 “1D Look-Up Table” 224

14 Steering System 225
14.1 Overview 225
14.1.1 Steering Interface 226
14.1.2 Steer by Angle 228
14.1.3 Steer by Torque 229
14.1.4 Steering-in-the-Loop test bench 230
14.2 General Steering System Parameters 231
14.3 Steering System “Static Steer Ratio” 232
14.4 Steering System “Dynamic Steer Ratio” 233
14.5 Steering System “Pfeffer” 235
14.5.1 Overview 235
14.5.2 The Mechanical Module 236
14.5.3 The Power Assistance Module 238
14.5.4 Steering System “Pfeffer” Parameters 244
14.6 Steering System "User C-code / Simulink Plug-in / FMU" 258
14.7 User Accessible Quantities for Steering Systems 258

15 Powertrain 259
15.1 Overview 259
15.2 PTControl 260
15.2.1 Overview 260
15.2.2 PTControl Interface 261
15.2.3 Interface to Brake Control 268
15.2.4 Operation states 268
15.2.5 Interpretation of gas pedal position 274
15.2.6 Mechanical energy management - Strategy modes 277
15.2.7 Battery management 294
15.2.8 PTControl Models 296
15.3 Control Units 298
15.3.1 Engine control unit (ECU) 298
15.3.2 Motor control unit (MCU) 302
15.3.3 Transmission control unit (TCU) 305

15.3.4 Battery control unit (BCU) 319
15.4 Drive Sources 322
15.4.1 Engine 322
15.4.2 Starter motor 341
15.4.3 Electric motor 343
15.4.4 Clutch 350
15.4.5 Gearbox 362
15.5 Driveline 377
15.5.1 Coupling 388
15.6 Power Supply 396
15.6.1 Power supply model 396
15.6.2 Battery 403
15.7 Powertrain Models 408
15.7.1 Overview 408
15.7.2 Powertrain model interface 409
15.7.3 User Accessible Quantities for Powertrain 412
15.7.4 Powertrain model "Generic" 413
15.7.5 Powertrain model "Parallel Hybrid" 415
15.7.6 Powertrain model "Axle Split Hybrid" 419
15.7.7 Powertrain model "Power Split Hybrid" 423
15.7.8 Powertrain model "Serial Hybrid" 426
15.7.9 Powertrain model "Electrical" 430
15.7.10 Powertrain model "OpenXWD" 433

16 Brake 449
16.1 Overview 449
16.2 Brake Interface 450
16.2.1 User Accessible Quantities for Brake 452
16.3 Brake model "Hydraulic" 453
16.3.1 Hydraulic Brake Control 453
16.3.2 Hydraulic Brake System 458
16.3.3 User Accessible Quantities for Brake model "Hydraulic" 460
16.4 Hydraulic Brake System "Pressure Distribution" 461
16.4.1 Overview 461
16.4.2 Hydraulic Brake System "Pressure Distribution" Parameters . 462
16.5 Hydraulic Brake System "HydESP" 464
16.5.1 Overview 464
16.5.2 Brake circuit configuration 466
16.5.3 Brake booster 468

16.5.4 Master Cylinder 476
16.5.5 Wheel brakes 478
16.5.6 Volume elements in general 481
16.5.7 Wheel brake cylinders 482
16.5.8 Low pressure accumulator 484
16.5.9 Attenuators (damper chambers) and line volumes 485
16.5.10 Hydraulic pump (return pump) 486
16.5.11 Valves and Connecting Lines in General 489
16.5.12 Solenoid valves 491
16.5.13 Proportional Solenoid Valves 493
16.5.14 Dynamic Solenoid Valves 495
16.5.15 Pilot valve with check valve and pressure limiting valve 501
16.5.16 Suction valve 503
16.5.17 Check Valve of the Low Pressure Accumulator 504
16.5.18 Temperature of the Brake Fluid 504
16.5.19 Parking Brake 507
16.5.20 User Accessible Quantities for Hydraulic Brake System "HydESP" 507

17 Tire 508
17.1 Overview 508
17.1.1 Tire Model with Contact Point Interface (CPI) 510
17.1.2 Tire Model with Standard Tire Interface (STI) 512
17.1.3 Overview of Tire Model Calculation in CarMaker 514
17.2 General Tire Parameters 515
17.2.1 Contact Point Modification 516
17.3 Tire Model "IPGTire" 517
17.3.1 Parameters 517
17.3.2 Approximation of the tire measurement data 519
17.4 Tire Model "RealTime Tire" 520
17.4.1 Rolling Resistance 522
17.4.2 Importing Tire Measurements 523
17.5 Tire Model "Magic Formula" 525
17.5.1 The basics of Magic Formula 525
17.5.2 Tire Description 527
17.5.3 Scale factors 531
17.5.4 Vertical force dependent on tire inflation pressure 535
17.5.5 Effective tire rolling radius 537
17.5.6 Slip computation 538

17.5.7 Longitudinal force (pure longitudinal slip): 539
17.5.8 Lateral force (pure sideslip): 542
17.5.9 Aligning Torque (pure sideslip) 547
17.5.10 Longitudinal Force (combined slip) 552
17.5.11 Lateral force (combined slip) 554
17.5.12 Aligning Torque (combined slip) 557
17.5.13 Overturning Couple 558
17.5.14 Rolling Resistance Torque 560
17.5.15 Transient Behavior 562
17.5.16 Turn Slip and Parking 565
17.5.17 Gyroscopic Couple 570
17.5.18 Friction coefficient 570
17.5.19 User Accessible Quantities 570
17.6 Tire Model "Tame Tire" 571
17.6.1 User Accessible Quantities 574
17.7 Tire Model "MF-Tyre/MF-Swift 6.2" 575
17.8 Tire Model "MF-Tyre/MF-Swift 7.2" 576
17.9 Tire Model "FTire" 578

18 Trailer Model 580
18.1 Overview 580
18.2 General Trailer Parameters 581
18.3 Mass Geometry 582
18.3.1 Overview 582
18.3.2 Parameters 583
18.3.3 Trailer with Twin Tires 583
18.4 Suspension Kinematics and Compliance 584
18.4.1 Sleeve Axle 585
18.4.2 Crank Axle 585
18.4.3 Semi Trailing Arm Axle 586
18.4.4 Additional External Movement 586
18.5 Suspension Force Elements 588
18.6 Hitch 588
18.6.1 Overview 588
18.6.2 Additional Parameters for Hitch “Ball” 590
18.6.3 Additional Parameters for Hitch “Trapezoid” 590
18.6.4 Additional Parameters for Hitch “Ball with Friction” 591
18.6.5 Additional Parameters for Hitch “Ball with Damping” 591

18.6.6 Additional Parameters for Hitch "Fifth Wheel" 592
18.7 Brake 593
18.7.1 Brake model “Overrun” 593
18.7.2 Brake model “Overrun with Friction” 594
18.7.3 Brake model “Hydraulic” 595
18.8 Aerodynamics 596
18.9 User Accessible Quantities for Trailer 596

19 Power Flow Calculation 597
19.1 Introduction 597
19.2 Definition 597
19.3 Parameters 598
19.4 PowerDelta 598
19.4.1 Overview 598
19.4.2 PowerTrain Delta 599
19.4.3 Brake Delta 603
19.4.4 Tire Delta 603
19.4.5 Suspension Delta 605
19.4.6 Aerodynamic Delta 606
19.4.7 Trailer Delta 608
19.4.8 Gravitation Delta 609
19.4.9 Chassis Inertia Delta 609
19.5 PowerStore 610
19.6 PowerLoss 611
19.7 User Accessible Quantities 611
19.8 Manipulation of Power Flow Calculation 611

20 Sensors 612
20.1 Introduction 612
20.1.1 Ideal Sensors for Rapid Prototyping/Function Development . . 612
20.1.2 High Fidelity Sensors for Function Development/Testing 613
20.1.3 Raw Signal Interfaces for Component Development/Testing . 613
20.1.4 Overview 613
20.2 Inertial Sensor 614
20.3 Slip Angle Sensor 615
20.4 Object Sensor 616
20.4.1 Introduction 616
20.4.2 Integration Levels 617
20.4.3 Object List 620

20.4.4 Calculation class 622
20.4.5 Target Selection 624
20.4.6 Overview of Object Sensor Calculation 628
20.4.7 Parametrization of Object Sensor 629
20.4.8 User Accessible Quantities 633
20.4.9 Visualization in IPGMovie 633
20.5 Free Space Sensor 634
20.5.1 Introduction 634
20.5.2 Parametrization of Free Space Sensor 634
20.5.3 User Accessible Quantities 636
20.6 Free Space Sensor Plus 637
20.6.1 Introduction 637
20.6.2 Functional description 637
20.6.3 Point filters 638
20.6.4 Parametrization of Free Space Sensor Plus 639
20.7 Traffic Sign Sensor 642
20.7.1 Introduction 642
20.7.2 Parametrization of Traffic Sign Sensor 642
20.7.3 User Accessible Quantities 644
20.8 Line Sensor 645
20.8.1 Introduction 645
20.8.2 Detection of the Lines 646
20.8.3 Parametrization of Line Sensor 648
20.8.4 User Accessible Quantities 650
20.9 Road Sensor 651
20.9.1 Introduction 651
20.9.2 Theory of the sensor prevision 651
20.9.3 Parameterization of Road Sensor 655
20.9.4 User Accessible Quantities 657
20.10 Collision Sensor 658
20.10.1 Introduction 658
20.10.2 Parametrization of Collision Sensor 658
20.10.3 User Accessible Quantities 660
20.11 Radar Sensor 661
20.11.1 Introduction 661
20.11.2 Detection Based on Signal-to-Noise Ratio 661
20.11.3 Processing Effects 665
20.11.4 User Accessible Quantities 666

20.11.5 Parametrization of Radar Sensor 668
20.12 Global Navigation Sensor 676
20.12.1 Introduction 676
20.12.2 Calculation 676
20.12.3 Error Modelling in CarMaker 677
20.12.4 Visibility Constrains due to Traffic 678
20.12.5 Parametrization of the Global Navigation Sensor 679
20.12.6 User Accessible Quantities 681
20.13 Ultrasonic Raw Signal Interface 682
20.13.1 Introduction 682
20.13.2 Included Effects 683
20.13.3 Wave Propagation 685
20.13.4 Processing 686
20.13.5 Object Model Requirements 687
20.13.6 User Accessible Quantities 687
20.13.7 Parametrization of the Ultrasonic RSI 687
20.14 Radar Raw Signal Interface 694
20.14.1 Introduction 694
20.14.2 Included Effects 694
20.14.3 Wave Propagation 696
20.14.4 Object Model Requirements 697
20.14.5 User Accessible Quantities 698
20.14.6 Parametrization of the Radar RSI 698

21 Traffic 702
21.1 Overview 702
21.2 Interface 704
21.3 General Parameters 706
21.4 Motion model 712
21.4.1 Pedestrian 712
21.4.2 Course 712
21.4.3 Ball 712
21.4.4 4Wheel 713
21.4.5 2Wheel 715
21.5 Maneuver 716
21.5.1 Maneuver Reference 716
21.5.2 General Maneuver Parameters 717
21.5.3 Longitudinal Motion 719

21.5.4 Lateral Motion 723
21.6 Autonomous Driving 726
21.6.1 Cruising 726
21.6.2 Following 727
21.6.3 Lane Change 728
21.6.4 Longitudinal Vehicle Dynamics 729
21.7 Free Motion 730
21.8 Input From File 730
21.8.1 Longitudinal Maneuver 731
21.8.2 Lateral Maneuver 731
21.8.3 Free Motion 732
21.9 User Accessible Quantities for Traffic 732

22 Geographic Coordinate System (GCS) 733
22.1 Introduction 733
22.2 Projection onto the road plane 734
22.2.1 GCS reference point 734
22.2.2 Theory of projection method *FlatEarth* *735*
22.2.3 Theory of projection method *GaussKrueger* *736*
22.3 User Accessible Quantities 738

23 Special Modes 739
23.1 Standby Mode 739
23.2 SlotCar Mode 739
23.3 Pylon Detection 741

24 User Accessible Quantities 743
24.1 General 744
24.1.1 TCPU 744
24.1.2 TGPU 744
24.1.3 Misc 745
24.2 Environment 746
24.3 Driving Maneuvers 747
24.4 Driver 748
24.5 Vehicle 750
24.6 Vehicle Control 752
24.6.1 General 752
24.6.2 ACC-Controller & AccelCtrl 753
24.6.3 Generic Longitudinal Control 753

24.6.4 Generic Lateral Control 753
24.7 Car 754
24.7.1 Car Convenience Quantities 754
24.7.2 Car 755
24.7.3 CarFlex 763
24.7.4 Elastic mounted Engine 763
24.8 MBS Suspension 764
24.8.1 Bushing 764
24.8.2 McPherson suspension 765
24.8.3 McPherson extended suspension 766
24.8.4 Fourlink suspension 768
24.8.5 Fourlink extended suspension 770
24.8.6 Twistbeam suspension 771
24.8.7 Twistbeam extended suspension 772
24.8.8 Double wishbone suspension 773
24.8.9 Double wishbone extended suspension 775
24.9 Steering System 776
24.9.1 Steering System -General 776
24.9.2 Steering System - Pfeffer 776
24.10 Tire 778
24.10.1 Tire Model Tame Tire 778
24.10.2 Tire Model Magic Formula 5.2 and 6.1 779
24.10.3 Tire Model MF-Tyre/MF-Swift 6.2 779
24.10.4 Tire Model MF-Tyre/MF-Swift 7.2 780
24.10.5 Tire Model FTire 780
24.11 Brake 782
24.11.1 Brake - General 782
24.11.2 Brake - Hydraulic 782
24.12 Powertrain 785
24.12.1 General 785
24.12.2 Powertrain Control (PTControl) 785
24.12.3 Engine Control Unit (ECU) 787
24.12.4 Motor Control Unit (MCU) 787
24.12.5 Transmission Control Unit (TCU) 787
24.12.6 Battery Control Unit (BCU) 788
24.12.7 Engine 788
24.12.8 Integrated Starter Generator / Electric Motor 789
24.12.9 Clutch 789

24.12.10 GearBox 790
24.12.11 Driveline 791
24.12.12 Power Supply 791
24.13 Power Flow Calculation 793
24.13.1 PowerDelta 793
24.13.2 PowerLoss 794
24.13.3 PowerStore 794
24.14 Sensors 795
24.14.1 Inertial Sensor 795
24.14.2 Slip Angle Sensor 795
24.14.3 Object Sensor 796
24.14.4 Free Space Sensor 797
24.14.5 Traffic Sign Sensor 798
24.14.6 Line Sensor 799
24.14.7 Road Sensor 800
24.14.8 Collision Sensor 801
24.14.9 Radar Sensor 801
24.14.10 Global Navigation Sensor 803
24.14.11 Ultrasonic RSI 805
24.14.12 Radar RSI 805
24.15 Trailer 806
24.15.1 General 806
24.15.2 Trailer Load 810
24.16 Traffic 811
24.16.1 General 811
24.16.2 Traffic object 811
24.17 Traffic Light 813

25 Start Conditions 814
25.1 Overview 814
25.2 Vehicle Start Configuration 814
25.3 Configure Vehicle Start Conditions 815
25.4 Software Interface 816

A Data Storage Result File Formats 818
A.1 IPG erg-files 818
A.1.1 Type 2 erg-files 818
A.1.2 Type 1 erg-files 821
A.1.3 Content of the accompanying Infofile 823

A.1.4 Measurement Data Format (MDF) 826

B Traffic Sign File Format 827

B.1 Traffic sign file format 827

B.1.1 Traffic sign infofile entries 827

C Traffic Signs Overview 832

C.1 France (FRA) 832

C.2 Germany (DEU) 833

C.3 Japan (JPN) 835

C.4 United Kingdom of Great Britain and Northern Ireland (GBR) 837

C.5 United States of America (USA) 838

Chapter 1
Introduction

This is the CarMaker Reference Manual. It contains descriptive information about the modules and models implemented in the software package including their interfaces and the list of the User Accessible Quantities.

1.1 General Remarks

CarMaker tries to keep as close as possible to conventions and naming of ISO 8855 2011, modified (which corresponds to DIN 70000).

This applies amongst other issues to axis systems, kinematics of the sprung mass, forces and moments, suspension, vehicle response and wheels and tires.

All parameters and quantities are specified in SI-quantities unless otherwise stated:

Table 1.1: SI Units used with CarMaker

Quantity	Name	Symbol
Time	second	s
Length	meter	m
Angle	radian (one turn = 2*π)	rad
Mass	kilogram	kg
Inertia	kilogram meter squared	$kg*m^2$
Force	Newton	N
Torque	Newton meter	Nm
Stiffness	Newton per meter	N/m
Rotational Stiffness	Newton meter per radian	Nm/rad

Two famous exceptions are:

- Speed in [km/h] for some input parameters
- Pressure in [bar] for the corresponding output quantities

1.2 CarMaker Axis Systems

In the virtual world of CarMaker different axis systems for different purposes are used. They are used to simplify calculation and parametrization for CarMaker objects (including signals and variables) and to be able to represent different points of views for CarMaker objects.

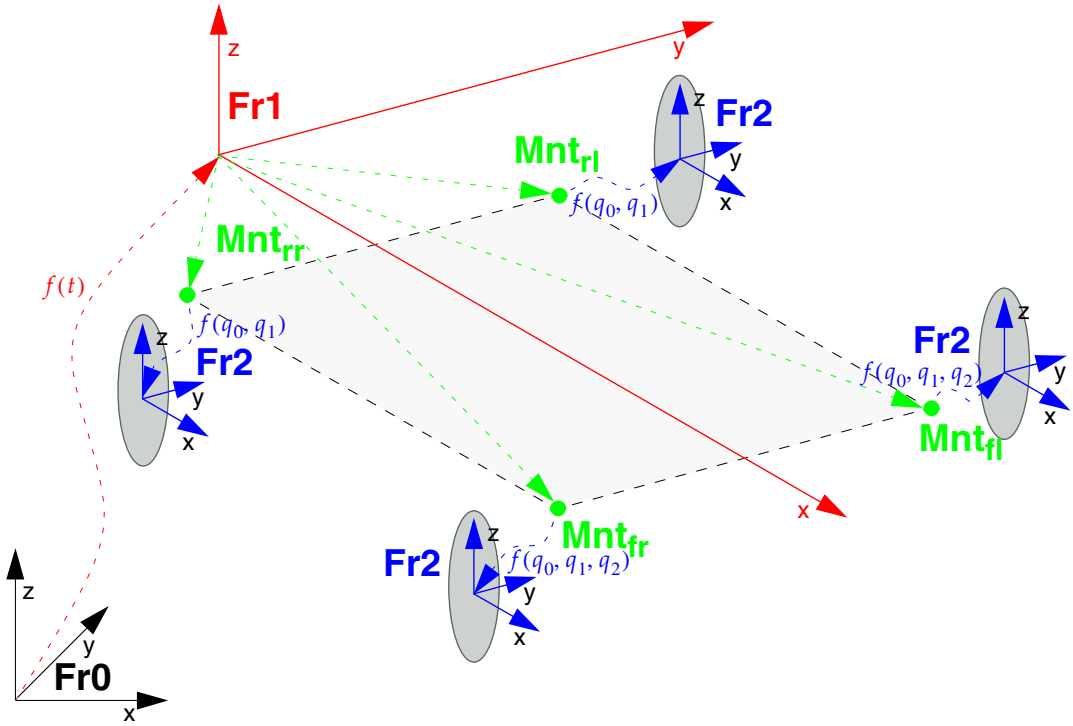


Figure 1.1: CarMaker coordinate systems

Frame Fr0

The CarMaker inertial axis system is called `Fr0` (pronounced: frame zero). This is the earth fixed origin of the 'virtual world'.

`Fr0` is defined as follows:

- (O) is the origin, (X), (Y), (Z) are the 3 axis.
- (O, X, Y) is the horizontal driving plane (road).
- (Z) is directed upwards (mathematically: $(X) \times (Y)$).
- The position of any point, if not mentioned explicitly otherwise, is expressed in Fr0.

Frame Fr1

Moving objects in the virtual world are based on their own accompanied axis system which is called `Fr1`. This axis system is fixed to the moving object. This means that the axis system *performs all movements* of the attached object like *translations* and *rotations*.

`Fr1` is defined as follows:

- (X) points in forward driving direction.
- (Y) points to the left.
- (Z) is directed upwards (mathematically: $(X) \times (Y)$).
- In case of a vehicle no part of the outer skin is situated *behind* the (O, Y, Z)-plane. At least one point of the outer skin has a vanishing X-coordinate. (see section 1.3)

- In case of a trailer no part of the outer skin is situated *before* the (O, Y, Z)-plane. (see section 1.3)

Frame Fr2 (carrier axis system)

For every wheel there is a mountpoint (`Mnt`) defined within the Fr1 system. This is the center of reference of a `Fr2` axis system attached to this mount-point. Mount-points are pure translations (X,Y,Z) from the Fr1 axis system. They are fixed to the Fr1 system.

There are functional dependencies (suspension kinematics and compliance) how `Fr2` is orientated relatively to its mount-point. There are three generalized coordinates (q_0, q_1, q_2) for the movement of each `Fr2` axis system. Usually q_0 stands for compression and q_2 for steer influence; q_1 stands for compression of the opposite wheel.

`Fr2` is defined as follows:

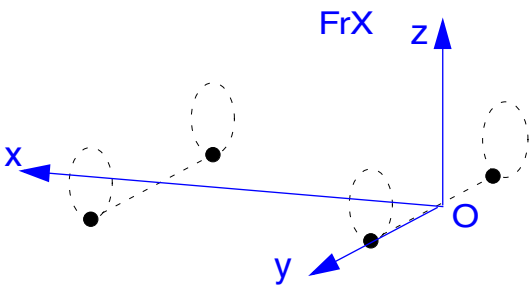
- Rules apply to any of the wheel carriers.
- (O) is the center of the wheel. It is in the wheel plane.
- (X) points in forward driving direction.
- (Y) is along the wheel spin axis. Vector points to the left.
- (Z) is directed upwards (mathematically: $(X) \times (Y)$).
- Initially (all coordinates of `Fr2` are zero) `Fr2` is parallel to `Fr1`.
- (O, X, Z) is the wheel plane.

Frame FrX

The frame `FrX` (pronounced: cross frame) represents a road surface axis system. The O-X-Y-plane approximately describes the current orientation of the road surface. Like `Fr1` this is a accompanied axis system as well and moves uniformly to `Fr1`.

`FrX` is defined as follows:

- (O) is located in the middle between the two road surface contact points of the rear wheels.
- (X) is orientated from (O) to the middle of the two road surface points of the front wheels.
- (Y) is orientated along the connection from the rear left to the rear right road surface contact point.
- (Z) is oriented upwards (mathematically: $(X) \times (Y)$).



Frame FrD

The `FrD` (pronounced: design-frame) is a parallel axis system to Fr1 with different origin. It is used to specify geometry input coordinates which are based on a different origin than `Fr1` without recomputing them. See section 1.3.2 'Guidelines defining CarMaker Objects' on page 20.

`FrD` is defined as follows:

- (O) is arbitrary.
- (X) is parallel to `Fr1` (X) axis.
- (Y) is parallel to `Fr1` (Y) axis.
- (Z) is parallel to `Fr1` (Z) axis.

1.3 CarMaker Geometry Input

To define new objects (vehicles, trailers) a good understanding of the CarMaker principles and conventions is needed to obtain the expected results. The information provided in this section is very useful for this task.

1.3.1 Conventions defining CarMaker Objects

As a convention CarMaker objects have to be defined with their origin of Fr1 at designated positions. This is important because CarMaker assumes the origin at those designated places for every object. The following table shows the conventions where the origin for which objects has to be:.

Table 1.2: Origins of CarMaker Fr1 Objects

Type of Fr1	Origin
Vehicle	Origin for the vehicles Fr1 is the hindmost point of the vehicle projected on road level.
Trailer	Origin for the trailers Fr1 is the hitch point (foremost point) of the trailer projected on road level. (Trailer objects have negative X-coordinates).

1.3.2 Guidelines defining CarMaker Objects

To define a new CarMaker object, geometry data has to be provided. Geometry data e.g. is needed for body mass, mount-points for (carrier-)frames, and additional loads (inclusive trimloads and engine mass).

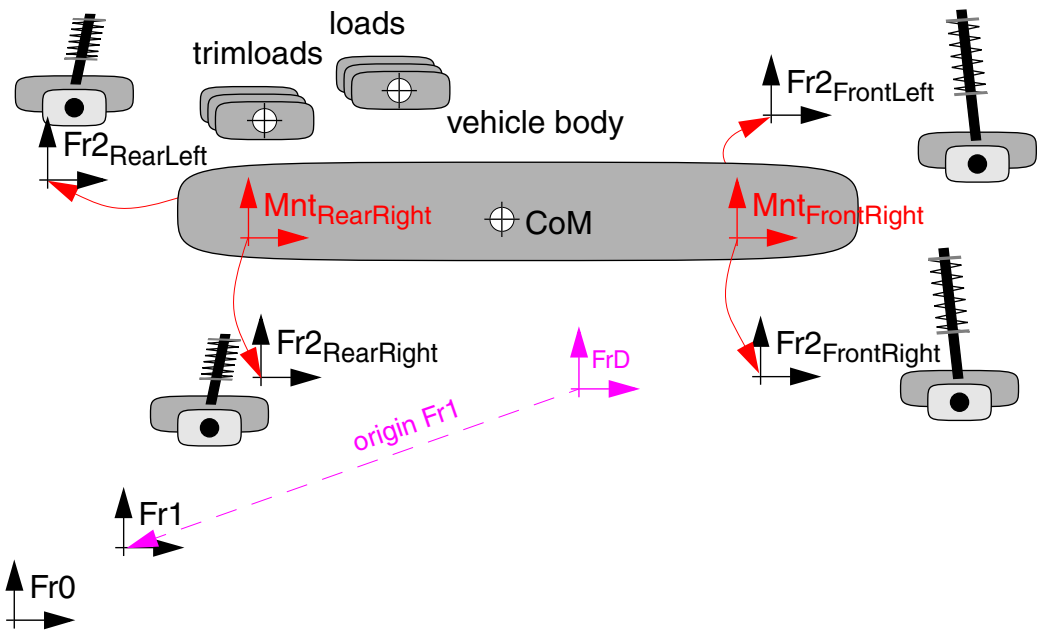


Figure 1.2: CarMaker geometry input for a vehicle

The position and orientation of any point/frame is given in the design configuration ('drawing sheet of the designer of the car'). As shown in Figure 1.2 the axis `FrD` (called design-frame) is a parallel axis system to Fr1 with different origin.

The configurations specified in `FrD` must be linked to the `Fr1` by a vector pointing to the origin of `Fr1` (expressed in `FrD` coordinates).

In the special case where FrD ≡ Fr1 the link vector is the zero vector.

Design configuration not necessarily has to be the same as the equilibrium configuration!

1.4 CarMaker Naming Conventions

To ensure maximum readability, it is tried to keep the notation as consistent and self-explanatory as possible.

1.4.1 CarMaker Subsystems

The following rules to name CarMaker subsystems apply:

Model Abbrev.	Description
Brake	Brake subsystem
Brake.Hyd.CU Brake.Hyd.Sys	Hydraulic brake control Hydraulic brake system
Car	Car subsystem
Car.Road	Road sensor in the middle of the front axle
DM	Maneuver control subsystem (DrivMan)
Driver	Driver subsystem
Env	Environment subsystem
FST	FailSafeTester
IO	IO subsystem
Log	Log subsystem
PT	Powertrain subsystem
PT.Control PT.ECU PT.BCU PT.MCU PT.TCU	Powertrain control unit Powertrain engine control unit Powertrain battery control unit Powertrain motor control unit Powertrain transmission control unit
Pwr	Power flow calculation
Sensor	Sensor subsustem
Sensor.Collision Sensor.FSpace Sensor.GNav Sensor.Inertial Sensor.Line Sensor.Object Sensor.Road Sensor.SAngle Sensor.TSign	Collision sensor Free space sensor Global navigation sensor Inertial sensor Line sensor Object sensor Road sensor Slip angle sensor Traffic sign sensor
Steer	Steering subsystem
TCPU	Monitoring of CPU time consumption
Traffic	Traffic object subsystem
TrfLight	Traffic light subsystem
Time	CarMaker timer
Tr	Trailer subsystem
VC	VehicleControl subsystem
Vhcl	Vehicle means subset of Car, Motorcycle, Truck subsystem

1.4.2 Classification of Quantities

With the following classifications, groups of quantities are denoted. Those groups denote points of interest like the center of mass of components or single components and their significant quantities.

The letters	... in general stand for
Aero	Aerodynamic related quantities
Buffer, Buf	Suspension buffer/bumper related quantities
C	Carrier related quantities
Camber	Camber (wheel) related quantities
Clutch	Clutch related quantities
Con	Quantities related to *connected body*: center of mass of the rigid vehicle body including vehicle body, engine, trimloads and loads
Damp	Damper (shock absorber) related quantities
DL	Driveline related quantities
Engine	Engine related quantities
Fr0, Fr1, Fr2, FrX	Quantities related to this Frame (= axis system)
Gearbox	Gearbox related quantities
Gen	Quantities related to the generalized body: center of mass of generalized body including all masses of the connected body including masses of wheel carriers and wheels. (“Car.Gen”, “Tr.Gen”)
Gen	Powertrain Generic related quantities (“PT.Gen”)
Hitch	Trailer hitch related quantities
HydESP	HydESP brake model related quantities (“Brake.HydESP”)
Jack	Jack related quantities
Load	Extra loads of the vehicle
PoI	Point of interest related quantities
Spring	Suspension spring related quantities
Stabi	Anti-Roll-Bar stabilisator related quantities
Steer	Steer subsystem related quantities
Valve	Brake hydraulic valves related quantities
Virtual	Virtual forces and torques related quantities
W	Wheel related quantities

1.4.3 Meaning of Abbreviations

CarMaker uses the following abbreviations for the naming of quantities.

The letters	... in general stand for
0, 1, 2, 3,..	Numbering
_0, _1, _2	Postfix for quantity expressed in `Fr0`, `Fr1`, `Fr2`
_ext	External
_tot	Total
a	Acceleration
Align	Aligning movement
Axle	Suspension left, right, front, back
C	Center
Diff	Differential
Distance, Dist	Distance
DL	Driveline
F	Front
FL, FR, RL, RR	Front left, front right, rear left, rear right
Frc	Force
Hori	Projection of a quantity to the O-X-Y-plane
l	Length
l	Lateral
LongSlip	Long slip (tangential) of wheel
Man	Maneuver
MC	Master brake cylinder
muRoad	Friction coefficient
No	Number
P	Tire contact point with track
p	Pressure
q	Flow (hydraulic)
q, q0, q1	Generalized coordinates
r	Rotation
R	Rear
Radius	Radius
Rate	Angular velocity
res	Resultant
rot	Rotation angle
rotv	Rotation angle velocity
Slip	Slip (lateral) of wheel
Spd	Speed
SS, SideSlip	Sideslip of vehicle
SW	Switch
t	Time, translation
T	Tire
T	Temperature
T, Time	Time
tau	Angle of attack of the wind

The letters	... in general stand for
Trq	Torque
v, vel	Velocity
WB	Wheel brake
x, y, z	Coordinate directions

Chapter 2

CarMaker Parameter Files

2.1 Infofile parameter format

CarMaker stores most of its parameters in so called Infofiles. Infofiles are basically plain ASCII files with a specific syntax.

Empty lines, lines containing only whitespace, and lines starting with '#' are ignored. The parameter entries come in two flavours:

- The parameter name followed by a '=' character and one or more values up to the end of the line.
- The parameter name followed by a ':' character. The parameter value is stored in the following lines, each line starting with a <tab> character.

The order in which parameter entries appear in the file is in principle irrelevant, since parameters are accessed by name, not by position. Nevertheless parameters are usually arranged according to a certain scheme with the intent of better legibility for human readers.

2.2 Common parameter entries

2.2.1 File identification

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 parameter is called "FileIdent". Together with the accompanying entries "FileCreator" and "Description" it is usually placed at the very beginning of a parameter file.

FileIdent = CarMaker-*KindString VersionId*

Syntax **CarMaker-<ModelClass>[-<Kind>] <Version ID>**

<ModelClass>	Tire, Brake, Car, TestRun,...
<Kind>	Optional, for submodels with referenced parameters, e.g. separate files for brake model parameters or engine torque diagram.
<VersionID>	Version of parameter file, whole-numbered, positive

Example

```
FileIdent = CarMaker-PTEngine-4WD 2
```

Known FileIdents

Among others, CarMaker supports the following FileIdents:

Infofile	FileIdent
TestRun	CarMaker-TestRun <VersionID>
TestSeries	CarMaker-TestSeries <VersionID>
Tire	CarMaker-Tire-<Kind> <VersionID>
Car	CarMaker-Car <VersionID>
Suspension (kinematics and compliance)	CarMaker-SuspKnC <VersionID>
Trailer	CarMaker-Trailer <VersionID>
Brake	CarMaker-Brake-<Kind> <VersionID>
ECUParameters	CarMaker-ECUParameters <VersionID>
SimParameters	CarMaker-SimParameters <VersionID>

Description :
DescriptionText
...

Description which may be displayed in parameter browsers for further information about the file, e.g. its purpose.

FileCreator = ***String***

Contains information about the software which wrote the file, possibly with a software version number, time and date.

2.2.2 Model kind

The kind key is used to specify a submodel (e. g. a specific powertrain model).

Prefix*.Kind = *KindStr VersionId

KindStr is the characteristic name of submodel, or variation of submodel. *VersionId* is used to support different versions of parameter sets for the same model (compatibility, parameter convertion etc.).

Example

```
PowerTrain.Engine.Kind = Mapping 2
```

2.2.3 Inclusion of Infofiles

It is possible to split the content expected from one Infofile into several files. To reference one or more files inside one Infofile the following command can be used (multiple times):

!include "*filename*"

The included file has to follow the Infofile syntax. However, FileIdents (see section 2.2.1 'File identification') must not be used in the included file(s). The referenced file itself can be located on top level of your current CarMaker project directory. If located elsewhere, the relative path is required:

```
!include "Data/Vehicle/MyAdditions"
```

An included file may also reside in a data pool. This process is entirely transparent for an application reading the Infofile, i.e. the application will only see the combined infofile data as if coming from a single big file; this also applies to the CarMaker GUI.

2.3 Selected TestRun parameters

2.3.1 Vehicle and trailer loads

VehicleLoad.<i>.pos =
VehicleLoad.<i>.mass =
VehicleLoad.<i>.l =
VehicleLoad.<i>.mounted = ***MountFrame***

Masses to setup TestRun mass contribution of the vehicle. Loads are mounted on *MountFrame* and can be moved while simulation is running (for example by changing their position by DVA). Their position is given in frame Fr1.
MountFrame can be `Fr1`, `Fr1A` *or* `Fr1B`, *default is* `Fr1` or `Fr1A`.
<i> := 0, 1, 2, 3

TrailerLoad.<i>.pos =
TrailerLoad.<i>.mass =
TrailerLoad.<i>.l =

Masses to setup TestRun mass contribution of the trailer. <i> := 0, 1, 2, 3. For details, see parameter `VehicleLoad`.

2.3.2 Random number generator

RandomSeed

Optional. Several modules (like sensors, traffic) use the *rand* function for random behaviour. The seed value parameter for the *rand* function is by default -1: in this case the seed value is generated internally dependent on current clock time and logged to the session log for the testrun reproduction, if at least one module uses the *rand* function with this seed value.
Besides the value can be set to each positive integer value.

Example `RandomSeed = 1512063718`

Chapter 3

CarMaker Configuration

CarMaker reads TestRun independent settings from the following configuration files::

File	Purpose
`SimParameters`	General settings.
`ECUParameters`	Parameters and calibration settings for ECUs used with a test stand.
`OutputQuantities`	List of quantities for data storage.
`Archive`	TestRun archiving parameters.

The files are in Infofile format (for details see the previous chapter). They are looked for in the *`Data/Config`* subdirectory of a CarMaker project directory.

Several instances of each file may exist in parallel, with a hostname appended to the filename. When CarMaker is looking for one of the above mentioned files, the file containing the name of the host CarMaker is currently running on has priority over its unspecific counterpart.

Example

```
Data/Config/SimParameters
Data/Config/ECUParameters
Data/Config/ECUParameters.rt1
```

3.1 SimParameters

The `SimParameters` file contains parameters controlling general, Testrun-independent behaviour of the simulation program. The file is reread at the beginning of each simulation.

All parameters listed below are optional., their default value is indicated.

3.1.1 General application parameters

FirstInit.TestRun = *TestRunName*

TestRun to initialize the application the first time, before a test run is started. Default: “” to use internal defaults.

TestRunEnd.DVA_ReleaseAll = *bool*

if set to true (1) DVA write access to quantities stops at the end of each TestRun. At the end of a test run, all active direct variable write access can be closed.
Default: 0, don’t release all, keep write access active.

TSInterval = *nCycles*

Specifies how often timestamps are taken in order to update the *SimCore.TCPU.XXX* variables (*TCPU.XXX* quantities in the data dictionary) used for performance measurement.

For CarMaker/HIL the default value is 1, i.e. time is measured each cycle. For the standalone variants of CarMaker the default value is 19 (as of version 2.1.12), i.e. time is measured once in 19 cycles.

3.1.2 Cycle timing

SimParam.DeltaT= *value*

Sets the cycle time used by CarMaker in seconds. Default: 0.001.
Please note, that the built-in IPG models have been validated for simulation with the default cycle time of 1ms. Using a different cycle time may influence the quality of simulation results and as such the setting is recommended only for special use cases.

3.1.3 Cycle timing (CarMaker/HIL)

Cycle.tCTViolationWarn = *DeltaT*

A measured cycle time exceeding this value (i.e. a cycle overrun) will lead to a warning message. The value may be specified as an absolute value in seconds or as a percentage of the cycle time (with a trailing percent sign %).

Default value is 130% of the cycle time, i.e. 0.0013 s for a cycle time of 1 ms.

Cycle.tCTViolationErr = *DeltaT*

A measured cycle time exceeding this value (i.e. a cycle overrun) will lead to an error message and a running simulation will be aborted. The value may be specified as an absolute value in seconds or as a percentage of the cycle time (with a trailing percent sign %).

Default value is 200% of the cycle time, i.e. 0.002 s for a cycle time of 1 ms.

Cycle.maxCTViolation = *value*

Default 1000.

Cycle.MeasuredDT = *bool*

If set to 1, use the measured cycle time as the step size passed on to the simulation framework, the vehicle model etc. for calculations. Using the measured cycle time was the default setting in CarMaker versions up to and including 4.5.2. In later versions the configured cycle time is used instead, which brings a more reliable, deterministic behavior of maneuver control, better reproduceable results and better compatibility with CarMaker/Office and CarMaker/HIL on other platforms.

Default 0 (i.e. use the configured cycle time).

This parameter has an effect only on CarMaker/HIL for XENO and ETAS.

3.1.4 Monitoring Realtime Conditions (CarMaker/HIL)

RTGuard.Mode = *Mode*

Defines how to react on Switches to Secondary mode (realtime not guaranteed). Possible modes are:

- *summary*: If switches to secondary mode are detected during a TestRun, one single warning will be printed to the session log after the simulation ends.
- *warn*: Whenever a switch to secondary mode is detected, a warning will be sent to the session log.

- *error*: Whenever a switch to secondary mode is detected, an error message will be sent to the session log.

Default mode is *summary*.

RTGuard.Verbose = *bool*

If set to 1, whenever a switch to the secondary mode is detected and reported to the session log, also print a stack back trace of the thread which caused the domain switch.

There is no effect, if *RTGuard.Mode* is *summary*.

Default 0.

RTGuard.maxLogs = *value*

Sets the maximum number of messages, if switches to secondary mode are reported to the session log during TestRun.

There is no effect, if *RTGuard.Mode* is *summary*.

Default 10.

3.1.5 Data Storage

DStore.OutputQuantities = *MyOutputQuantitiesFName*

Configuration file for selecting quantities to be written to file. Filename relative to Data/Config/<HostName>. Default name is “OutputQuantities”.

DStore.OutPath = *PathSpec*

Sets the default value used for construction of the data storage path. Defaults to `%o/%r/%D/%t_%T%?_s`. A detailed documentation about all %-macros which may be used in the path can be found in the description of ScriptControl’s *SetResultFName* command in the Programmer’s Guide.

DStore.OutSubDir = *DirectoryName*

Directory to store simulation results. If *DirectoryName* is empty, the current date is used as the directory name, formatted as <year><month><day>. Default “”. This entry also sets the value of the %t macro used for construction of the whole data storage path.

DStore.OutFNameWithTime = *WithTime*

Append simulation start time to the result file name. Default 1.

DStore.BufSize_kB = *Size_in_kBytes*

Size of data collecting buffer.

DStore.MaxFileSize_MB = *Size_in_Megabytes*

Maximum result file size. Upon reaching this limit, data storing will automatically stop. Default value is 2048, i.e. 2 GB; a negative value like e.g. -1 means "no limit".

3.1.6 Session / TestRun commands

SessionCmds:
rtexpr
rtexpr
...

Commands (i.e. Realtime Expressions) to be executed permanently, from the start of the simulation program, and also when in idle state. Will be reinitialized during TestRun initialization if the *SimParameters* file was modified.

TestrunCmds:
rtexpr
rtexpr
...

Commands (i.e. Realtime Expressions) to be executed while a simulation is running. Will be reinitialized during each TestRun initialization. These commands can be seen as supplementary to the *Maneuver Commands* defined in a TestRun.

3.1.7 Vehicle Model

Vehicle.Tire.TrqT2W_withEffRollRadius =*bool*

Specifies if the tire reaction moment, used for the wheel speed integration in powertrain module, is calculated with loaded radius or with effective rolling radius.

This parameter is valid for all tires of car, trailer and mcycle model and is independent on tire model (STI or CPI interface). By default, the car and trailer use the effective rolling radius (=1) due to correct power flow balance. Mcycle uses still the loaded radius (=0) as default.

Vehicle.Limit.Roll = *AngleLimit_deg*

Maximal roll angle. Default: 80 deg.

Vehicle.Limit.Pitch = ***AngleLimit_deg***

Maximal pitch angle. Default: 80 deg.

3.1.8 Tire Model MF-Tyre/MF-Swift

TNO MF-Tyre/MF-Swift 6.2

TNO_MF62.LibPath.<arch> =*Path*

If another library, than the one provided within the CarMaker installation, should be used. Specifies the path of the TNO tire model dynamic library.

The *<arch>* extension is an optional specifier to define which library should be used on which architecture. In case there are multiple keys defined, the key and its corresponding library which matches to the currently used architecture, has priority. If there is no key with the appropriate architecture provided, the key without any architecture extension will be used.
Possible extensions are: "win32", "win64".

Example

```
TNO_MF62.LibPath.win32 = C:/tmp/win32/TNO_DelftTyre.dll
TNO_MF62.LibPath = C:/tmp/TNO_DelftTyre.dll
```

TNO_MF62. LogGenMsg = ***bool***

If active all general messages are printed out, else only the warning and error messages.

TASS MF-Tyre/MF-Swift 7.2

TASS_MF.License = ***License***

This key is only needed if the environment variable MADLIC_LICENSE_FILE is not set.

This key holds the MF-Tyre/MF-Swift license, and must be set to <port>@<license server>. Where <port> is the FLEXlm port which is 26000 by default and <license server> is the host name of the computer serving the license (e.g. 26000@myServer).

TASS_MF.LibPath.<arch> =*Path*

If another library, than the one provided within the CarMaker installation, should be used. Specifies the path of the TASS tire model dynamic library.

The *<arch>* extension is an optional specifier to define which library should be used on which architecture. In case there are multiple keys defined, the key and its corresponding library which matches to the currently used architecture, has priority. If there is no key with

the appropriate architecture provided, the key without any architecture extension will be used.
Possible extensions are:"win32", "win64", "xeno".

Example
```
TASS_MF.LibPath.xeno = /tmp/mfswift_carmaker.so
TASS_MF.LibPath = C:/tmp/mfswift_carmaker.dll
```

TASS_MF. LogGenMsg = ***bool***

If active all general messages are printed out, else only the warning and error messages.

3.1.9 Maneuver Control

Road.VhclStartPos_TrailerMin = ***MinStartPos_m***

Minimal vehicle road start coordinates when driving with a trailer. Default 10 m.

Shifting

DrivMan.AutoShift.dtDontShift = ***delatT***

Default 0.2 s.

DrivMan.AutoShift.GasReduction = ***Reduction***

Default 0.8.

DrivMan.AutoShift.dt_declutch = ***DeltaT***

Default 0.1 s.

DrivMan.AutoShift.dt_keepclutch = ***DeltaT***

Default 0.1 s.

DrivMan.AutoShift. dt_enclutch= ***DeltaT***

Default 0.5 s.

DrivMan.VelCtrl.tClutchRelease = ***value***

Default 1.0 s.

3.1.10 Start of TestRun

SimStart.TimeLimit = ***TimeLimit***

Maximal duration of state SCState_SimStart. Default: 240 s.

DrivMan.Start.ExtInp_SteerVel = ***SteerVelMax_deg***

Used for external inputs maneuvers to build up the steering angle at start of input.

3.1.11 Vehicle Reload

Vehicle.Reload = *ModeStr*

Specifies when the vehicle model (including all car parts, brake, powertrain and trailer) should be reloaded and with this re-initialized. By default in CarMaker/Office, the model is always initialized. In CarMaker/HIL, the model is only initialized if one of the vehicle or trailer infofiles has been modified. This functionality is not supported with CarMaker for Simulink.

The following modes are available:

- *Always*: All vehicle parts are reloaded and initialized at each simulation start (default for non-HIL).
- *Auto*: All vehicle parts are only reloaded and initialized if one of the vehicle model infofiles has been modified (default for HIL).
- *Skip*: Skip the initialization of all vehicle parts.

For the *Auto* mode, the functionality checks for modification of:

- vehicle, trailer and tire parameters in testrun infofile
- vehicle and trailer load parameters in testrun infofile
- Named Values and Key Values (TestManager)
- vehicle, trailer and any infofiles which are used as external files for model selection (for example: tire infofile, brake infofile)
- the return value in the Model Manager function *ParamsChanged.* Each model has to define in the register function the *ParamsChanged* function. This function should check if parameters from another parameter file (like the tire property file in the models *RTTire* and *MFTire*) or the model library itself has been modified.
  If there are no further parameter files or model library, which can be changed, the *ParamsChanged* function should be set to

  `ParamsChanged=ParamsChanged_IgnoreCheck.`

  If the function is not set, the *Auto* functionality always reloads the vehicle.

  For following models the *ParamsChanged* function is not set:
  - tire model *IPGTire*
  - powertrain model *AVL CRUISE*
  - any *FMU* models
  - any *RTW Plugin* models

If the vehicle is not initialized before the next simulation, it is recommended to end and start the simulation with zero velocity (standstill) to avoid discontinuities in the velocity signals.

Vehicle.Reload.Debug = *bool*

Debug flag to log if the vehicle is reloaded/initialized or not. Default: 0.

For Vehicle.Reload = Always no log message will be displayed.

3.1.12 End of test run

GetIdle.TimeLimit = ***TimeLimit***

Default 200 s.

GetIdle.Skip = ***bool***

Default: 0 for HIL, 1 for non-HIL.

GetIdle.BodyDecel = ***value***

Body deceleration in the *GetIdle* simulation state. Default: 4 m/s^2.

GetIdle.RotVelDecel = ***value***

Body rotational deceleration in the *GetIdle* simulation state. Default: 20deg/s^2.

DrivMan.Idle.OperatorActive = ***bool***

Specifies if the vehicle operator is activated or not during the idle phase. Default: 1.

DrivMan.Idle.OperatorState = ***string***

Specifies the target operation state during the idle phase.
The following values are possible for default vehicle operator *IPGOperator*:

- *absent*: Absent, vehicle left (default for non-HIL)
- *poweroff*: Power off
- *poweracc*: Power accessory
- *poweron*: Power on (default for HIL)
- *drive*: Drive
- *userX*: User defined state x (x beginning with 0; *user0* is the first value after *drive*)
- *keep:* Keep the actual target operation state for the idle phase

3.1.13 Pylon detection

PylonDetect.Log = ***KeyList***

Activates the logging while preparation phase (Key=Pre) and while simulation phase (Key=Sim).

Example `PylonDetect.Log = Sim`

Example `PylonDetect.Log = Pre Sim`

PylonDetect.nSPNotesMax = ***value***

Specifies the maximal number of scratchpad notes in one TestRun. Default: 20.

3.1.14 QuantAudit

QuantAudit.MaxCount = ***value***

Specifies the maximum number of scratchpad notes for a single QuantAudit job. Default: 16.

QuantAudit.DontStore = ***bool***

A non-zero value means, that Scratchpad entries created by QuantAudit jobs are not to be stored in simulation results files. Default: 0.

3.1.15 Scratchpad

Scratchpad.MaxTypes = ***value***

Specifies the maximum number of scratchpad note types. Default: 128.

Scratchpad.PreallocatedNotes = ***value***

Specifies the maximum number of preallocated scratchpad notes (on CarMaker/HIL only).

If, during a TestRun, the number of preallocated scratchpad notes is exceeded, a warning message will be logged to the session log. Exceeding this limit, may affect realtime conditions due to dynamic memory allocation.

Default: *Scratchpad.MaxTypes * QuantAudit.MaxCount*.

3.1.16 Geographic coordinate system

GCS.Traffic.Active = *bool*

Activates the calculation of global position, expressed in geographical coordinate system for the traffic objects. Default: 0.

3.1.17 ADTF Interface

ADTFParam.FName = *filename*

The configuration file for the ADTF interface needs to be configured with this key. The configuration file (e.g. Test123) needs to be saved under */Data/Config/* in the CarMaker project folder.
In case there is a file with appended hostname in the file name (e.g. Test123.host1), then this will be chosen preferably to the one that has no hostname extension (e.g. Test123).

3.1.18 CRUISE Interface

Cruise.LicenseFile.<arch> = *License*

This key holds the CRUISE license, e.g.:
License = “lib/License.dat”
Absolute as well as relative paths are allowed. If this key is not given the flexlm-License Manager takes care about where to find a license file.

The *<arch>* extension is an optional specifier to define which library should be used on which architecture. In case there are multiple keys defined, the key and its corresponding library which matches to the currently used architecture, has priority. If there is no key with the appropriate architecture provided, the key without any architecture extension will be used.
Possible extensions are: "linux", "linux64", "win32", "win64", "xeno"

Example
```
Cruise.LicenseFile.linux = /tmp/License.dat
Cruise.LicenseFile = lib/License.dat
```

Cruise.LibraryPath.<arch> = *LibPath*

This key holds the path and name of the CRUISE library.
Default (Windows/Linux): *LibPath* = “lib/cruise_m.dll/so”

The <arch> extension is an optional specifier for different architectures. If there is no key with the appropriate extension given, the key without any extensions will be used.
Possible extensions are: "linux", "linux64", "win32", "win64", "xeno".

Example
```
Cruise.LibraryPath.win32 = C:/tmp/win32/cruise_m.dll
Cruise.LibraryPath = lib/cruise_m.dll
```

Cruise.ProjectPath_other= *Path_other*
Cruise.ProjectPath_win32= *Path_win32*

The keys *Cruise.ProjectPath_win32* and *Cruise.ProjectPath_other* are used to adapt the path given by *PowerTrain.Cruise.Project* in the Vehicle Infofile or the *CRUISE project* entry in the Vehicle Data Set GUI to the path definition of the actual operating system.
In case the application is running on a linux system and the *CRUISE project* is given in Windows notation, then it is tested if the left side of the *CRUISE project* matches the path given by *Cruise.ProjectPath_win32*. If they match, the matching part will be replaced by the path given in *Cruise.ProjectPath_other*, otherwise an error is generated. If the application is running on a Windows system and the *CRUISE project* is given in the path notation of a other system, then it is checked wheter the left side of the project path matches *Cruise.ProjectPath_other*, and in case of a match be replaced by the path given by *Cruise.ProjectPath_win32*. If the *CRUISE project* is given in the notation of the actual OS these keys have no effect.

Please note, instead of the *CRUISE project* (*.prj) you can also load *CRUISE_M* files (*.tsk).

Example:

CRUISE project: *C:/AVL/Cruise/v2015/projects/ManFWD/CarMaker/ManFWD.prj*
Path_win32: *C:/AVL/Cruise/v2015/projects*
Path_other: */projects*

Then the CRUISE project path is changed to:

CRUISE project: */projects/ManFWD/CarMaker/ManFWD.prj*

If a relativ project path ought to be used just don't set these keys in the *SimParameters* file.

Default: empty string

Cruise.SuppressOutputFiles = *SuppressOutputFiles*

This parameter allows to switch on/off the writing of result files needed by the CRUISE post-processing.
Default: 1.

Cruise.CM-DVA.SkipQuants = *SkipQuants*

This key allows to switch off the automatic generation of the AVL_CRUISE quantities
Default value *SkipQuants* = 0

3.1.19 GT-SUITE Interface

GT.License = *License*

This key is only needed if the environment variable GTISOFT_LICENSE_FILE is not set.
This key holds the GT-SUITE license, and must be set to <port>@<license server>. Where <port> is the FLEXlm port which is 27005 by default and <license server> is the host name of the computer serving the license (e.g. 27005@myServer).

GT.HomeDir.<arch> = *GTIHOMEPath*

This key is only needed if the environment variable GTIHOME is not set.
This key holds the path to the GT-SUITE installation, e.g.
GTIHOMEPath = “/opt/GTI”.

The <arch> extension is an optional specifier for different architectures. If there is no key with the appropriate extension given, the key without any extensions will be used.
Possible extensions are: "linux", "linux64", "win32", "win64", "xeno".

Example
```
GT.HomeDir.linux = /opt/GTI
GT.HomeDir = C:/Program Files/GTI
```

GT.Lib.<arch> = *LibPath*

This key holds the path and name of the GT-SUITE library. It’s only needed if you want to manually overwrite the default path to the GT-SUITE library. If the environment variable GTIHOME is defined CarMaker will use the library in the GT-SUITE installation folder. If GTIHOME is not defined CarMaker:
Default (Win/Linux/XENO): *LibPath* = “lib/libgtlink_dp.so”, "lib/gtlink_dp.dll", "lib/libgtcfd.so".

The *<arch>* extension is an optional specifier to define which library should be used on which architecture. In case there are multiple keys defined, the key and its corresponding library which matches to the currently used architecture, has priority. If there is no key with the appropriate architecture provided, the key without any architecture extension will be used.
Possible extensions are: "linux", "linux64", "win32", "win64", "xeno".

Example
```
GT.Lib.linux = /tmp/lib/libgtlink_dp.so
GT.Lib = lib/gtlink_dp.dll
```

GT.sldur= *value*

Define the simulation duration for the GT-SUITE solver. Default: -1

GT.RT.ErrLvl= *value*

Define the messaging level for the GT-SuiteRT solver. If set to anything else than "0" realtime isn’t guaranteed anymore. Only set this parameter, if you have problems with the solver and realtime isn’t needed. Default: 0

3.1.20 Video Data Stream Client

VDS.MovieHost

Host running IPGMovie. Default: localhost

VDS.MoviePort

TCP/IP port for listening. Default: 2210

VDS.Verbose

Activates logging output. Default: 0.

3.1.21 ADAS RP

ADASRP.Host=<hostname>

Please replace *hostname* by the name of your PC.

ADASRP.Port=21323

Sets the communication port between CarMaker and ADAS RP.

ADASRP.ReceiveInterval = *<time>*

This parameter defines the observation interval between received messages, default: 0.001s.

ADASRP.SendInterval = *<time>*

Thisis the interval between two sent messages for position and velocity, default: 0.1s.

3.1.22 Functional Mock-up Interface (FMI)

FMU.Logging.ToFile = *bool*

A non-zero value means, that logging output of any FMU will be sent to file `FMU.log` in the project directory, not to the standard CarMaker log file. Default: 0.

FMU.Logging.ExcludeCategs:
category
category
#category
...

FMU log messages of one of the log categories listed under this key will be filtered out and thrown away, i.e. they will not go to any log file. List entries starting with a hash (#) character will be ignored.

This key is applied to *all* FMUs; an identically named parameter in an FMU's `.pluginfo` file exists for FMU-specific filtering. Both the global and the FMU-specific category exclude lists will be considered when filtering messages.

FMU.LocationFormat:
<FMU> <n>
<FMU> <n>
#<FMU> <n>
...

When an FMU is being instantiated, it gets passed the URI of its location. In CarMaker this is always a string like

file:///some/directory/with/the/unpacked/fmu

Because the exact URI format was never specified by FMI 1.0, some ambiguity arises concerning the number of slashes at the beginning, especially when dealing with MS Windows paths containing drive letters. An FMU may prove to be quite picky about what it accepts as a valid location URI, so the FMU.*LocationFormat* entry was created in order to allow overriding the default number of slashes (n=3) for certain FMUs.

Example: With a list entry like `MyBrake_FMU 1` , for all instances of MyBrake_FMU the above URI would become

file:/some/directory/with/the/unpacked/fmu

List entries starting with a hash (#) character will be ignored.

FMU.UseDVA=*<0/1>*

If set to 1 (default), this key allows FMU output signals to execute a DVAwrite when being connected to a writable User Accessible Quantity. This key is used to restore the default behavior until and including CarMaker 6.0.4, in which a DVAwrite from an FMU output signal was not possible.

3.1.23 CarMaker for Simulink

CM4SL.ReadCMParameter.FailIfAccessDenied = *bool*

Determines the behavior of *Read CM Parameter* blocks in a Simulink model when trying to access a key from an encrypted (secured) infofile and that particular key was not explicitly listed as "open (unlocked)" upon encryption.

CM4SL.ReadCMParameter.FailIfAccessDenied = 0
If a default value for the key was specified in the block parameters, then that default value will be used. If no default value was specified, an "Access to secured data denied" error will be issued. This is the default behavior.

CM4SL.ReadCMParameter.FailIfAccessDenied = 1
Independently of whether a default value for the key was specified or not an "Access to secured data denied" error will be issued. This used to be the default behavior until and including CarMaker 5.1.2.

3.1.24 Generic Plug-in Model

GenPlugins.InAllSimStates = *value*

Specifies in which simulation state the generic plug-in model should be executed. Currently, the generic plug-ins is run in every simulation state, even in *Idle*. Its execution is stopped only during the (re)initialisation of this generic plug-in.

For each generic plug-in additional parameters can be set in order to define at which place in the CarMaker simulation cycle the generic plug-in is executed and under which conditions it is reinitialised. They can be found in Programmer's Guide section B.3 'General Parameters' on page 914.

To avoid the permanent execution of the plug-in, set the parameter to 0. It means, the model will only be called during simulation state *Simulate*. Default: 1.

3.1.25 GPU Sensors

GPUSensor.Timeout_s = *value*

Not used in CarMaker/HIL. Specifies the maximum number of seconds CarMaker will wait for a GPU Sensor's calculation result before disabling the sensor. Default: 2 s.

3.2 ECUParameters

In the `ECUParameters` file all input/output, hardware or ECU specific parameters are kept. Typical parameter categories include:

- Signal conditioning
- FailSafeTester configuration created by the FailSafeTester Configuration GUI; for details see the corresponding section in the User's Guide.

The file is reread at the beginning of each simulation.

3.3 OutputQuantities

This file keeps a list with all quantity names (one per line spelled like they can be looked up in IPGControl). All quantities denoted in this file are stored by the CarMaker´s `Data Storage` functionality. The file is reread at the beginning of each simulation.

Only one '*' wildcard per pattern can be specified.

DStore.Format = *format*

Result file format. "erg" = type 2 erg-file (default), "mdf4.1" = Measurement Data Format (MDF), "ascii" = ASCII format for plain text separated by tabulator.

DStore.dt.fast

Time step between two data vectors for data stream of kind *fast* (only available with *DStore.Format = mdf4.1*). Default 0.1 s.

DStore.dt.normal

Time step between two data vectors for data stream of kind *normal* (used for both *DStore.Format = erg / mdf4.1*). Default 0.01 s.

DStore.dt.slow

Time step between two data vectors for data stream of kind *slow* (only available with *DStore.Format = mdf4.1*). Default 0.001 s.

DStore.Quantities.fast:
<quantity name>
<quantity name
...

List of User Accessible Quantities saved with sample mode *fast* (only available with *DStore.Format = mdf4.1*).

DStore.Quantities.normal:
<quantity name>
<quantity name
...

List of User Accessible Quantities saved with sample mode *normal* for *MDF* storage mode. It is the only quantity list for storage mode *erg*.

DStore.Quantities.slow
<quantity name>
<quantity name
...

List of User Accessible Quantities saved with sample mode *slow* (only available with *DStore.Format = mdf4.1*).

3.4 Archive

Contains configuration parameters for TestRun archiving. The file is read by the CarMaker GUI. A complete description of all parameters can be found in the file itself.

Chapter 4

Environment

4.1 General Parameters

The parameters explained here apply for *all* environment models and are stored in testrun data file.

Env.Temperature = *Temperature*

Specifies the environment air temperature in degree Celsius ˚C at mean sea level. Default: 20 ˚C.

Env.AirDensity = *Density*

Specifies the environment air density in kg/m^3 at mean sea level. Default: 1.205 kg/m^3.

Env.AirPressure = *Pressure*

Specifies the environment air pressure in bar at mean sea level. Default: 1.013 bar.

Env.AirHumidity = *Humidity*

Specifies the environment air humidity in percent (0..100%). Default: 60%.

Env.SolarRadiation = *SolarRadiation*

Specifies the environment solar radiation in W/m^2. Default: 400 W/m^2.

Env.RainRate = *RainRate*

Specifies the rain rate in mm/h. Default: 0.0 mm/h.
This parameter currently has an influence on the damping calculation of the radar sensor only.

Env.VisRangeInFog = *Range*

Specifies the visual range in fog in m. Default is 1000.0 m.
This parameter currently has only influence on the damping calculation of the radar sensor.

Global wind

Env.Wind.Kind = *string*

Specifies the environment global wind model to be used:

`none`	No global wind
`const`	Constant wind, which is added to the road wind markers

Env.Wind.Velocity = *Velocity*

Specifies the global constant wind velocity in km/h. Default: 0 km/h.

Env.Wind.Angle = *Angle*

Specifies the global constant wind direction angle in deg. Zero means wind direction along the inertial x-axis. Default: 0 deg.

Env.StartTime.Year = *Year*

Specifies the year when the simulation begins. Default: 2014.

Env.StartTime.Month = *Month*

Specifies the month when the simulation begins. Default: 1.

Env.StartTime.Day = *Day*

Specifies the day when the simulation begins. Default: 1.

Env.StartTime.Hour = *Hour*

Specifies the time of day when the simulation begins (0 ... 24 hours). Default: 12 hours.

Env.StartTime.Min = *Minute*

Specifies the time of day when the simulation begins (0 ... 60 minutes). Default: 0 minutes.

Env.StartTime.Sec = *Second*

Specifies the time of day when the simulation begins (0 ... 60 seconds). Default: 0 seconds.

Env.StartTime.DeltaUTC = *Hours*

Specifies the difference to UTC when the simulation begins. Default: 0 hours.

Example `Env.StartTime.DeltaUTC = -3.5`

Env.GNav.Active = *bool*

Specifies if the satellites of the global navigation sensor should be simulated. Default: 0.

Env.Kind = ***KindStr*** ***VersionId***

Specifies which environment model is used. Possible models are:

Model Name	KindStr	Description
Generic	Generic	simple environment model

Example `Env.Kind = Generic 1`

4.2 Environment Model "Generic"

The model "Generic" calculates with a simplified approach the environment air temperature, air pressure and air density. Other environment parameters are hold constant.

4.2.1 Model description

Air temperature

The environment air temperature is calculated as follows

$$T = T_0 + T_{elev} + T_{sRoad} + T_{time} \quad \text{(EQ 1)}$$

with temperature for height above mean sea level zero T_0, temperature offset depending on elevation T_{elev}, temperature offset depending on road coordinate T_{sRoad} and temperature offset due to time of day T_{time}.

The temperature offset due to elevation is

$$T_{elev} = \Delta h \cdot c_{elev} \quad \text{(EQ 2)}$$

with height above mean sea level Δh and the temperature decrease coefficient c_{elev}.

The temperature offsets T_{sRoad} and T_{time} are stored in 1D-lookup tables.

Air pressure

The environment air pressure is calculated as follows

$$p = p_0\left(1 + \frac{\Delta h \cdot c_{elev}}{T_0}\right)^{5.255} \quad \text{(EQ 3)}$$

with air pressure for height above mean sea level zero p_0.

Air density

The environment air density is determined as follows

$$\varsigma = \frac{p}{R_S \cdot T} \quad \text{(EQ 4)}$$

with the specific gas constant R_S. Using this formula the air humidity is neglected.

4.2.2 Environment Model "Generic" Parameters

Env.Temp.Offset_Elev = *value*

Specifies the coefficient for temperature decrease depending on elevation in K/m. Corresponds to the parameter c_{elev} in (EQ 2). Default: -0.0065 K/m.

Env.Temp.Offset_sRoad.Amplify = *value*

Amplifies the output of the characteristic air temperature offset along road by a given factor. Default: 1.0.

Env.Temp.Offset_sRoad: *data*

One dimensional characteristic for the air temperature offset along road. Corresponds to the variable T_{sRoad} in (EQ 1). Negative road coordinate is not allowed.

Syntax Infofile table mapping with 2 columns
<road coordinate [m]> <temperature offset [K]>

Example
```
Env.Temp.Offset_sRoad:
0       0
100     0
102    -1
250    -1.5
252     0
260     0
```

Env.Temp.Offset_Time.Amplify = *value*

Amplifies the output of the characteristic air temperature offset due time of day by a given factor. Default: 1.0.

Env.Temp.Offset_Time: ***data***

One dimensional characteristic for the air temperature offset depending on time of day. Corresponds to the variable T_{time} in (EQ 1). The time value must be between 0 and 24 hours. The temperature values for 0 and 24 hours must be the same.

Syntax
Infofile table mapping with 2 columns
<time [hours]> <temperature offset [K]>

Example

```
Env.Temp.Offset_Time:
0     -1
6     -2
12     0
16    +2
20     0
24    -1
```

4.3 Material Parameters

The *Material Library* provides material specific information to GPU based sensor calculations (*Free Space Sensor Plus / Ultrasonic RSI / Radar RSI*). The parameters are specified in the file named `MaterialLib` located in the folder `/Data/Sensor`. To add or remove material definitions, please copy this file from the installation directory to the CarMaker project folder, again to the `/Data/Sensor` folder.

The material's name is the link between the 3D geometry used by IPGMovie and its physical properties defined in the *Material Library*. It is used for tagging the 3D geometry defined in the *.obj file with help of the adjunct *.mtl file. The *.mtl file needs to use the same material names as defined in the *Material Library* with exactly the same spelling (case sensitive!). Please find further information about the material definition in the 3D object files in the IPGMovie User Manual, section 4.3 'Defining and Using Material IDs for the GPU Sensors'.

Please note: In case there is no match between the materials defined in the *Material Library* and those used in the *.mtl file, the GPU sensors use a default material:

- the *RSI sensors* use *metal* as default
- the *Free Space Sensor Plus* will return a material ID of 0

FileIdent = **MaterialLib**

Materials.N = **int**

Specifies the number of different materials. Default: 18.
Do not forget to modify this key if materials are added or removed!

Materials.<no> = int

The numbering of the materials starts with 1. The number of the material corresponds to the material ID provided by the sensors.

Materials.<no>.Name = string

Name of the material which should be identical to the name defined in the *.mtl file.

Materials.<no>.Radar.Permittivity = double

Relative electric permittivity for electromagnetic waves at the considered frequency. Default values are selected for 77 GHZ.

Materials.<no>.Radar.Scattering = double

Standard deviation of the stochastic perturbation of the perfect electromagnetic wave propagation direction in degree. If the value is set to zero, no scattering will occur.

Materials.<no>.Radar.Scatter = bool

Determines whether the material acts as scattering center. Default: 0. If the material acts as scattering center (`.Scatter=1`) the `Scattering` value will define the maximum allowed deviation of the incident direction from the face normal.

Materials.<no>.USonic.Reflection = double

Relation of the outgoing and incoming sound pressure amplitude. If the value is set to zero, the sound pressure wave will be fully dissipated and no reflection occurs.

Materials.<no>.USonic.Scattering = double

Standard deviation of the stochastic perturbation of the perfect sound pressure wave propagation direction in degree. If the value is set to zero, no scattering will occur.

4.4 User Accessible Quantities

Please refer to section 24.2 'Environment' on page 746.

Chapter 5

Driving Maneuvers

5.1 Overview

A CarMaker testrun is a sequence of one or more maneuver steps, called “mini maneuvers”. The mini maneuver actions consist of

- longitudinal dynamics actions: accelerating, braking, gear shifting, ...
- lateral dynamics actions: steering
- additional actions, defined by a list of mini maneuver commands.

Parameters

The driving maneuvers and driving behavior can be configured

- in the test run (test run specific)
- in the vehicle model parameter set (vehicle specific)
- in the SimParameters parameter set (global).

A prefix is built up for each mini maneuver by the string “DrivMan.”, followed by the mini maneuver number. The keys of all parameters for each maneuver start with this prefix.

Example

```
DrivMan.<i>.<xyz> ...
```

TestRun Data Set

DrivMan.nDMan = ***NumberOfMiniMan***

This entry defines the number of minimaneuvers of this test run. The first maneuver is the maneuver 0 (zero).

Sensor.PoI.pos = ***x y z***

Optional. Vehicle point of interest. The state (position, velocity, ...) of this point is used to control the vehicle. This parameter overwrites the corresponding vehicle parameter, see section 'Sensor.PoI.pos = x y z'. Coordinates defined in Fr1. Default: Center point of all wheels; for the z-component the wheel diameter is taken.

Vehicle Data Set

DrivMan.nShift : ***GearNo_nUp_nDown_table***

Optional. Configures the auto shifting module of the speed controller. Each line contains the following values:

- Gear number *GearNo*, positive for forward, negative for backward gears
- engine speed to shift up, unit rpm.
- engine speed to shift down, unit rpm.

Sensor.PoI.pos = ***x y z***

Optional. Vehicle point of interest. The state (position, velocity, ...) of this point is used to control the vehicle. Coordinates defined in FrD. This parameter can be overwritten by test run parameters, see section 'Sensor.PoI.pos = x y z'. Default: Center of all wheels; for the z-component the wheel diameter is taken.

5.2 Longitudinal Dynamics Maneuvers

The following maneuvers are available:

- Drive with IPGDriver
- Stop with IPGDriver (Stop Vehicle)
- Drive backwards with IPGDriver
- Drive speed profile with IPGDriver
- Manual Control: Gas, Brake, BrakeLever (motorcycle), BrakePark, Clutch, Gear
- Speed Control

5.2.1 IPGDriver

Pre*.LongDyn = Driver *Vel_km/h

Based on driver model IPGDriver.

The Vel_km/h parameter is optional and specifies the cruising speed to be used during this maneuver. If not specified, the cruising speed from the global IPGDriver parameters will be used.

5.2.2 Stop Vehicle

Pre*.LongDyn = Stop *maxDecel_m/s*^{2} *StopDistance

Based on driver model IPGDriver.

5.2.3 Drive backwards

Pre*.LongDyn = Backward *maxAccel_m/s*^{2} *Vel_km/h

Based on driver model IPGDriver. The vehicle should stand still before beginning this maneuver.

5.2.4 Manual

***Pre*.LongDyn = Manual**
Pre.Controls* = *TrgVal* *tStart* *tBuildUp* *Mode

To handle a control the activity has to be defined by

- the target value of the control,
- the starting time relative to the beginning of the maneuver, when the activity has to be started

- the build up time, to reach the target value, and
- the mode, that the value is absolute or relative.

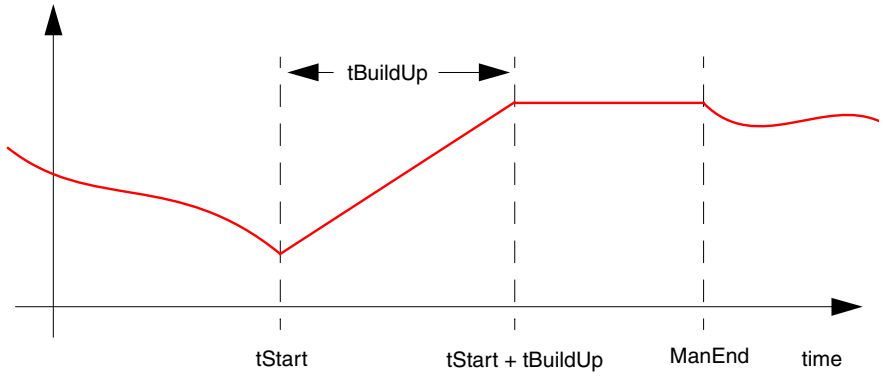


Figure 5.1 Manual Maneuver Control: Overwrite a Signal with an absolute value

Supported *Controls* are: “Gas”, “BrakeLever”, “BrakePedal”, “BrakePark”, “Clutch”, “Gear”. At the same time one or all controls can be handled. *tStart* and *tBuildUp* are floating point values. Modes *Mode* are “abs”, “offset”. The target value is hold as long as the maneuver is the active one.

The *Control* "Clutch" is only taken over in *DM.Clutch* in case of a manual gearbox kind. For the *Control* "Gear" there is a special treatment depending on vehicle gearbox kind. In case of a manual gearbox, "Gear" is taken over to the signal *DM.GearNo*. In case of no gearbox or a gearbox without Manumatic, "Gear" sets the signal *DM.SelectorCtrl*. In case of automatic gearbox with Manumatic (possible only for gearbox in combination with combustion engine) and "Gear">1, the *Manumatic mode* is activated by setting the selector control to the position M (*DM.SelectorCtrl=2*) and the target gear number is set in *DM.GearNo*.

5.2.5 Speed Control

Pre*.Long = VelControl *Vel_km/h TolVel_m/s Sensity PremEnd

Velocity control with a PID-controller. The minimal and maximal engine speeds for each gear are based on IPGDriver parameters.
The target velocity *Vel_kmh* can be manipulated while the maneuver via Direct Variable Access on the quantity *DM.v.Trgt*.

5.2.6 Speed Profile

Pre*.LongDyn = VelExtInput *restart shifting

Drive velocity profile vel(t), defined by CarMakers External Inputs module. Based on IPGDriver and its parameters (e.g. shifting time, the minimal and maximal engine speed for each gear).

restart: 1 to drive the velocity profile from its beginning, 0 to continue.

shifting: 1 the IPGDriver shifts the gears automatically during this maneuver; 0 no shifting

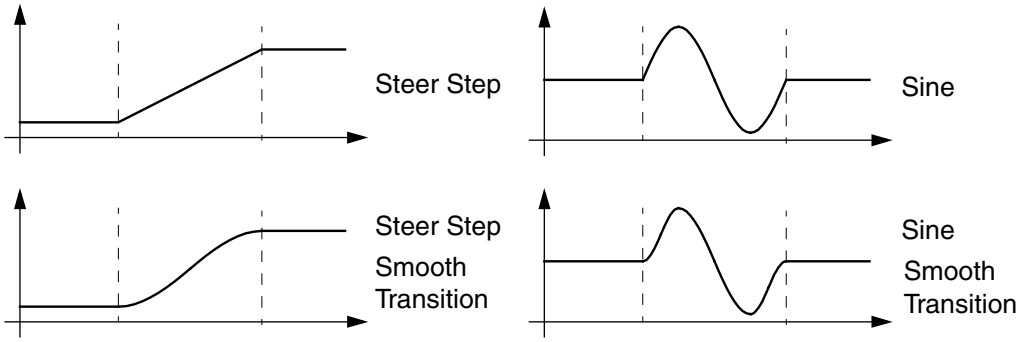
5.3 Lateral Dynamics Maneuvers

The following maneuvers are available:

- IPGDrive
- Sine steering (Sine and Sine Sweep)
- Steer step
- Simple steer control (Follow Course)

Steering Maneuvers

A smooth transition mode was added for Sine and Steer Step. The mode can be enabled with a checkbox in the CarMaker GUI's ***Maneuver*** dialog, under ***Lateral Dynamics***.



5.3.1 IPGDriver

***Pre*.LatDyn =**	**Driver**

Drive the course based on driver model IPGDriver.

5.3.2 Steer Step

***Pre*.LatDyn =**	**Step *Trgt tStart tBuildUp AR SM***
***Pre*.Rider =**	**Rider *Trgt tStart tBuildUp AR SM***

Target value *Trgt* of steer angle or the steer torque. The actuation starts at *tStart* relative to the mini maneuver begin and builds up the *Trgt* in *tBuildUp. AR* : abs, offset. *SM* : 0 for linear, 1 for smooth (sine). The "*Rider*" variant is for Motorcycle driving.

5.3.3 Sine Steering

Pre*.LatDyn = Sinus *Ampli TMode TPeriod tStart nPeriods Mode SM PhaseOffset

Ampli is steering amplitude in degree or steering torque. *TMode*: freq, time. *Mode* : abs, offset. *SM* : 0 for linear, 1 for smooth (sine).

5.3.4 Sine Sweep

Pre*.LatDyn = SinusSweep *Ampli0 Ampli1 TMode TPeriod0 TPeriod1 tStart nPeriods Mode SineMode

Steering starts with amplitude *Ampli0* and period time *TPeriod0* at the time tStart. While *nPeriodes* the amplitude is lineary changed to the end values *Ampli1*. The duration of periode is changed to *TPeriod1* accourding to the SineMode that is used. *TMode*: freq, time. *Mode* := abs, offset. *SineMode*: lin, exp

5.3.5 Simple Steering Control (Follow Course)

Pre*.LatDyn = SteerControl *Tolerance Sensitivity

Follow the road middle line taking lateral *Tolerance* into account. The control intensity is configured by *Sensitivity*. Used from SimParameters: DrivMan.Ctrl.LatDyn.LimitLow, DrivMan.Ctrl.LatDyn.LimitHigh.

5.4 Special Maneuvers

5.4.1 The last Mini Maneuver (Duration 0 Seconds)

If the last mini maneuver has the **duration 0 seconds**, it has a special meaning:

This maneuver is called at the end of the test run even if the test run is stopped before (by user, by an error, ...).

5.5 Vehicle Operator

All longitudinal and lateral maneuvers including the IPGDriver except the manual control maneuver expect that the powertrain of the vehicle is ready and is still in the operation state "drive".

A submodule of the module DrivMan (Driving Maneuvers) called *Vehicle Operator* has the task to reach a desired operation state (get the powertrain to power off or drive state), coming from the PTControl model inside the powertrain, by handling the vehicle elements like key, start-stop button, pedals, gear shifter in a defined manner and order (see Figure 5.2).

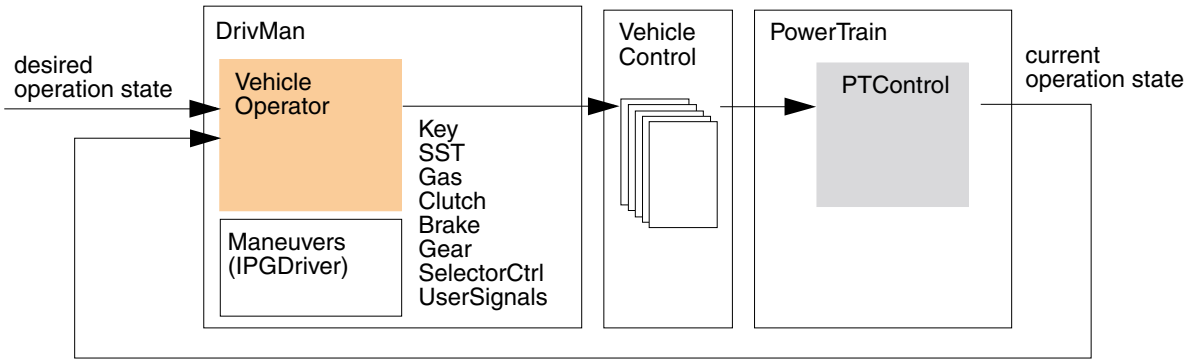


Figure 5.2: Interaction between *VehicleOperator* and *PTControl*

If the vehicle operator is activated and the current operation state differs from the desired state, the vehicle operator overwrites the maneuver outputs to reach the target state.

The vehicle operator can be activated and deactivated during simulation by setting directly the global C-variable *DrivMan.OperatorActive* or via DVA on the quantity *DM.OperatorActive*.

In same manner the target operation state can be modified during simulation by setting directly the global C-variable *DrivMan.OperationState_trg* or via DVA on the quantity *DM.OperationState_trg*.

DrivMan.Init.OperatorActive = *bool*

Specifies if the vehicle operator is activated or not during the preparation phase and at the beginning of the simulation. Default: 1.

DrivMan.Init.OperatorState= *string*

Specifies the target operation state at the beginning of the simulation.
Following values are possible for the default vehicle operator *IPGOperator*:

- *absent*: Absent, vehicle left
- *poweroff*: Power off
- *poweracc*: Power accessory
- *poweron*: Power on
- *drive*: Drive (default)
- *userX*: User defined state x (x beginning with 0; *user0* is the first value after *drive*)

DrivMan.VhclOperator.Kind = *KindStr VersionId*

Selection of vehicle operator model kind to use. The DrivMan components library provides following model to use.

ModelName	KindStr	Description
IPGOperator	IPGOperator	Generic vehicle operator, possible to start with start-stop button or ignition key and able to handle manual or automatic gearbox

The following table represents the input and output variables of the initialization interface struct *tVhclOperatorCfgIF*:

Input Variable	Unit	Description
StartEngineWithSST	bool	1: Powertrain is activated with start-stop button 0: Powertrain is activated with key only
GBKind		Gearbox kind (manual, automatic)

The following table represents the input and output variables of the evaluation interface struct *tVhclOperatorIF*:

Input Variable	Unit	Description
OperationState_trg		Target vehicle operation state
OperationState		Current vehicle operation state
OperationError		Vehicle operation error or warning
Velocity	m/s	Vehicle speed
Key		Vehicle key position
SST	bool	Activation start-stop button
Gas		Gas pedal position
Brake		Brake pedal position
BrakeLever		Brake lever position
BrakePark		Park brake position
Clutch		Clutch pedal position
SelectorCtrl		Gearbox selector control
GearNo		Gear position
Steering.Ang Steering.AngVel Steering.AngAcc Steering.Trq	rad rad/s rad/s^2 Nm	Steering wheel angle (velocity, acceleration) and steering wheel torque
UserSignal<*us*>		User defined signals *us*

Output Variable	Unit	Description
OperatorFinished		Flag if vehicle operator has finished (target reached)
Key		Vehicle key position
SST	bool	Activation start-stop button

Output Variable	Unit	Description
Gas		Gas pedal position
Brake		Brake pedal position
BrakeLever		Brake lever position
BrakePark		Park brake position
Clutch		Clutch pedal position
SelectorCtrl		Gearbox selector control
GearNo		Gear position
Steering.Ang Steering.AngVel Steering.AngAcc Steering.Trq	rad rad/s rad/s^2 Nm	Steering wheel angle (velocity, acceleration) and steering wheel torque
UserSignal<*us*>		User defined signals *us*

Vehicle Operator Model "IPGOperator

The vehicle operator model is able to start a powertrain with a start-stop button or with ignition key. For the detailed description of the both starting procedures please refer to section 15.2.4 'Operation states'.

DrivMan.VhclOperator.tWaitForNextState = *value*

Optional. Specifies the waiting time to start aiming the next operation state. Default: 0.5s.

DrivMan.VhclOperator.Steer2Zero.Grad = *value*

Optional. Specifies the steer angle velocity to center steering system to a predefined target position. Default: 30deg/s.

DrivMan.VhclOperator.Brake2Stand.v_lim = *value*

Optional. Specifies the vehicle velocity limit during simulation to decide for a proper braking to standstill and shutdown or for a emergency case. Default: 5km/h.

DrivMan.VhclOperator.Brake2Stand.v_BrakeEnd = *value*

Optional. Specifies the target vehicle velocity if braking until standstill. Default: 0.1km/h.

DrivMan.VhclOperator.Brake2Stand.Grad = *value*

Optional. Specifies the brake build-up gradient for braking until standstill. Default: 500/s.

DrivMan.VhclOperator.Brake2Stand.Brake = *value*

Optional. Specifies the maximum brake pedal position if braking until standstill.
Default: 0.7.

5.6 User Accessible Quantities

Please refer to section 24.3 'Driving Maneuvers' on page 747.

Chapter 6

VehicleControl

6.1 Introduction

The main idea of the VehicleControl module is to separate the connection between the DrivMan and the Vehicle module. The DrivMan module delivers the desired values for *Gas, Brake, Clutch, Steering Angle*, etc. The VehicleControl module decides in which manner the desired signals from DrivMan are transposed to the Vehicle module (PowerTrain, Brake, Steering, etc) according to requirements. Using this module any ECU could be integrated between the DrivMan and the Vehicle module to control the longitudinal and lateral dynamic motion, e.g.: ACC-Controller.

6.2 VehicleControl Interface

In the same manner like most submodules (Brake, PowerTrain, etc) a user specified VehicleControl model can be registered using the Model Manager mechanism.

The VehicleControl is called after the DrivMan and before the Vehicle module. Right before calling the VehicleControl model, all DrivMan signals are copied to the VehicleControl signals and hold as default values for further modifications. Then the user specified model is called to manipulate the VehicleControl signals according to the requirements. These VehicleControl signals are inputs for the Vehicle module.

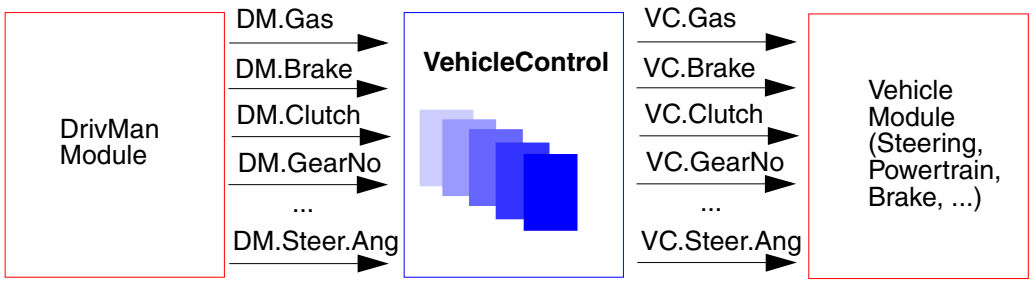


Figure 6.1: VehicleControl Interface

The VehicleControl module can call up to ten VehicleControl models one after another. The modificated VehicleControl signals by the first model are directly taken as inputs for the second model, and so on.

An included headerfile *VehicleControl.h* describes in detail the VehicleControl interface struct.

6.3 General Parameters

Following parameters can be specified in the TestRun or in the vehicle infofile. The parameters are read first from the TestRun infofile and, if not found, from the vehicle infofile.

VehicleControl.<i>.FName = *FileName* ***or***
VehicleControl.<i>.Kind = *KindString VersionNumber*

The parameter *FName* specifies the parameters file name of the VehicleControl model <i=0..9> including the model specific parameters. This parameters file should be located in *Project/Data/VehicleControl* and the header of the file must begin with:

```
#INFOFILE1 - Do not remove this line!
FileIdent = CarMaker-VehicleControl-KindString VersionNumber
```

Optionally the selected VehicleControl model <i=0..9> can be set directly by the parameter *Kind*. For this case all model specific parameters must be defined in the TestRun/vehicle file.

The CarMaker VehicleControl model library provides the following models:

ModelName	KindStr	Description
Acceleration Control + ACC	AccelCtrl	ACC controller with PI-controller for desired longitudinal acceleration
Generic Longitudinal Control	GenLongCtrl	Generic longitudinal controller with simplified AEB and FCW functions
Generic Lateral Control	GenLatCtrl	Generic lateral controller with simplified LKAS and LDW functions

Example `VehicleControl.Kind = AccelCtrl 1`

VehicleControl.Lights.Brake.SetManual = *bool*
VehicleControl.Lights.Reverse.SetManual = *bool*

Specifies if the light signal (brake, reverse) is linked to the corresponding vehicle control signal (pedal, gear position) or if the light signal can be set manually by user.
Default=0: light signal is set depending on the vehicle control signal.

6.4 ACC Controller

6.4.1 ACC Controller Model

CarMaker comes with a simple implementation of an ACC Controller. It controls the longitudinal accelaration of the vehicle by changing the position of the brake and gas pedal. If ACC is deactivated there is no manipulation of the pedal position by the controller. The controller distinguishes two cases:

- If there is no target detected, the velocity will be controlled.
- If there is a relevant target detected, the distance will be controlled.

Figure 6.1 represents the controller for a constant distance.

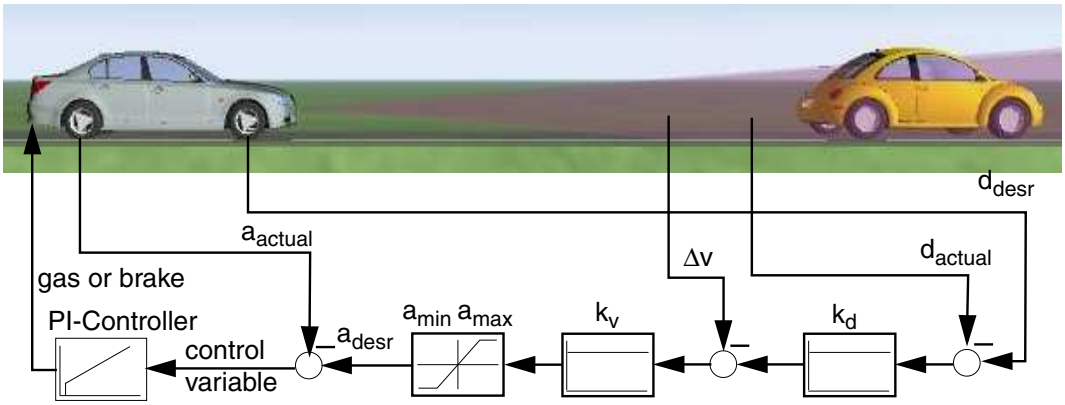


Figure 6.2: Closed loop of the ACC-Controller

6.4.2 ACC Controller & VehicleControl *AccelCtrl*

To modify the gas or brake pedal position depending on a desired longitudinal acceleration, a VehicleControl model *AccelCtrl* (Acceleration Controll) was implemented in CarMaker. This controller uses a PI-Controller, see Figure 6.3. The desired longitudinal acceleration can be calculated in different manners: using CarMakers built-in ACC-Controller, via Direct Variable Access or using a user implemented function:

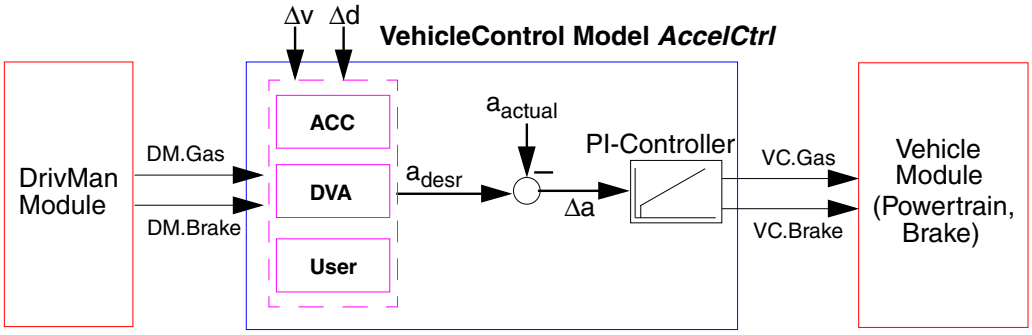


Figure 6.3: VehicleControl *AccelCtrl* with ACC Controller

6.4.3 Model Parametrization

Parameters for AccelCtrl

AccelCtrl.DesrAccelFunc = *FcnKind*

Specifies the function kind to calculate the desired acceleration for the *AccelCtrl*. Possible values are:

- *ACC*: The desired acceleration is calculated by CarMakers built-in ACC controller. This is the default.
- *DVA*: The desired acceleration is set using Direct Variable Access on quantity *AccelCtrl.DesiredAx*.
- *User*: The desired acceleration is calculated by a user implemented function, as shown in the following example:

Example

```
In User.c:
static void MyCalcAccel(double dt) {
    double Ax;
    Ax = ...;                //place here the controller algorithm
    AccelCtrl.DesrAx = Ax;
}

int User_Register (void) {
    Set_UserDesrAccelFunc(MyCalcAccel);  //register the user function
    return 0;
}
```

For all function kinds the acceleration control can be turned off by providing a desired acceleration of NOTSET=-99999 m/s^2. Then the control of gas pedal and brake is handed over to the DrivMan (maneuver control).

AccelCtrl.p = *pFactor*
AccelCtrl.i = *iFactor*

Specifies the parameters of the acceleration PI-Controller.
Default: p=0.001, i=1.0.

Parameters for the ACC-Controller

AccelCtrl.ACC.IsActive = *bool*

Activates the ACC-Controller. Default: 1 (=on).

AccelCtrl.ACC.RefObjectSensorName = *name*

Specifies the name of the reference ObjectSensor, which is used by the ACC-Controller. Default: *RadarL*.

AccelCtrl.ACC.BrakeThreshold = *value*

Specifies the Driver/DrivMan brake threshold beyond the ACC controller is being deactivated. Default: 0.2.

AccelCtrl.ACC.DesrTGap = *value*

Specifies the initial desired time distance to the target vehicle. Unit: [s]. Default: 1.8.

AccelCtrl.ACC.DesrSpd = *value*

Specifies the initial desired cruise speed. Unit: [m/s]. Default: vehicle start velocity if higher than 10km/h, else 100km/h.

AccelCtrl.ACC.DistMin = *value*

Specifies the minimal desired distance to the target vehicle. Unit: [m]. Default: 20.

AccelCtrl.ACC.AxMin = *min*
AccelCtrl.ACC.AxMax = *max*

Specifies the minimal and maximal allowed desired acceleration of the ACC controller. Unit: m/s^2. Default: AxMin=-2.5, AxMax=1.0.

AccelCtrl.ACC.DistCtrl.kd = *DistFactor*
AccelCtrl.ACC.DistCtrl.kv = *VelFactor*

Specifies the gain parameters for the distance controller of the ACC controller.
Default: kd=36.0, kv=2.0.

AccelCtrl.ACC.SpdCtrl.kv = *value*

Specifies the gain parameter for the velocity controller of the ACC controller.
Default: 13.0.

6.4.4 User Accessible Quantities

Please refer to section 24.6.2 'ACC-Controller & AccelCtrl' on page 753.

6.5 Generic Longitudinal Control

The simple *Generic Longitudinal Control* model contains two functionalities: *Autonomous Emergency Braking AEB* and *Forward Collision Warning FCW*, which are explained below.

6.5.1 Autonomous Emergency Braking (AEB)

The *Autonomous Emergency Braking (AEB)* system has the task to decelerate safely the vehicle to the velocity of the target object ahead. For this, the system compares the time-to-collision t_{tc} with a time-threshold-brake t_{tb} to decide if a braking intervention is required. In this simple generic model, these time values are calculated following the book *"Handbuch Fahrerassistenzsysteme, Winner, Hakuli, Wolf, 2009, chapter 33"*.

For a stationary or a very slow moving target object, the time-to-collision t_{tc} is calculated as follows:

$$t_{tc} = \frac{d}{v_{rel}} \qquad \text{(EQ 5)}$$

with relative distance d and relative velocity v_{rel} between the ego vehicle and the target object.

If the target object is decelerating, the time-to-collision is calculated as follows:

$$t_{tc} = \frac{\sqrt{v_{rel}^2 - 2 \cdot d \cdot D_{rel}} - v_{rel}}{D_{rel}} \qquad \text{(EQ 6)}$$

with relative deceleration D_{rel}.

The time-threshold-brake t_{tb} for a stationary target object is calculated in following way:

$$t_{tb} = \tau_B + \frac{v_{rel}}{2 \cdot D_{max}} \qquad \text{(EQ 7)}$$

with brake loss time τ_B.

For a non-stationary target object the time-threshold-brake is calculated with:

$$t_{tb} = \tau_B + \frac{v_{rel} + D_{rel} \cdot \tau_B}{2 \cdot (D_{max} - D_{obs})} \qquad \text{(EQ 8)}$$

with maximum allowed deceleration of ego vehicle D_{max} and the actual target object deceleration D_{obs}.

If $t_{tc} < t_{tb}$, the AEB system sets as target acceleration for the controller the maximum allowed deceleration D_{max}.

6.5.2 Forward Collision Warning (FCW)

The *Forward Collision Warning (FCW)* system has the task to warn the driver by different degrees of warning level if time-to-collision t_{tc} falls below the defined time threshold. The simple generic model supports two different warning levels, which are activated before the AEB reaction.

6.5.3 Model Parametrization

Parameters for AEB

AEB.SwitchedOn = *bool*

Switches the functionality of *Autonomous Emergency Braking* on or off.
Default: 1 (=on).

AEB.RefObjectSensorName = *name*

Specifies the name of the reference ObjectSensor, which is used by the AEB model.
Default: *FrontRadar*.

AEB.Decel_max = *value*

Specifies the maximum allowed deceleration (positive value) for the emergency braking (D_{max} from (EQ 8)). Default: 6m/s^2.

AEB.p = *pFactor*
AEB.i = *iFactor*

Specifies the parameters of the acceleration PI-Controller.
Default: p=0.001, i=3.0.

AEB.dist_min = *value*

Specifies the minimum distance to the vehicle ahead, under which the required deceleration is set to the maximum allowed deceleration value. Default: 5m.

AEB.t_wait = *value*

Specifies the time the vehicle continues braking after reaching standstill (v<0.2m/s).
Default: 5.0s.

AEB.tau_B = *value*

Specifies the loss time until the brake reacts (τ_B from (EQ 7)). Default: 0.2s.

Parameters for FCW

FCW.SwitchedOn = *bool*

Switches the functionality of *Forward Collision Warning* on or off.
Default: 1 (=on).

FCW.tau_warnLevel1 = *value*

If time-to-collision t_{tc} falls below the sum of FCW.tau_warnLevel1 and time-threshold-brake t_{tb}, the first warning level is set.
Default: 2s.

FCW.tau_warnLevel2 = *value*

If time-to-collision t_{tc} falls below the sum of FCW.tau_warnLevel1 and time-threshold-brake t_{tb}, the second warning level is set.
Default: 1s.

6.5.4 User Accessible Quantities

Please refer to section 24.6.3 'Generic Longitudinal Control' on page 753.

6.6 Generic Lateral Control

The simple *Generic Lateral Control* model contains two functionalities: *Lane Keeping Assist System LKAS* and *Lane Departure Warning LDW*, which are explained below.

6.6.1 Lane Keeping Assist System (LKAS)

The *Lane Keeping Assist System (LKAS)* has the task to support the driver to keep the vehicle in the middle of the lane for security and comfort aspects. For this, the system needs information like deviation angle and deviation distance to the ideal path (middle of the lane). In general, for the detection of the lane a camera-unit is used. In the model, for the determination of the middle of the lane, either a *Line Sensor* or a *Road Sensor* is used. Using the *Line Sensor*, the system requires two valid lines, either standard (white) or priority (yellow) lines. The latter are indicated by the key "ColorIdx = 10" inside their point list.

Figure 6.4 shows the approach of the steering assist torque calculation to support the driver. This simple generic model follows the description from the book *"Handbuch Fahrerassistenzsysteme, Winner, Hakuli, Wolf, 2009, chapter 34 and 35"*.

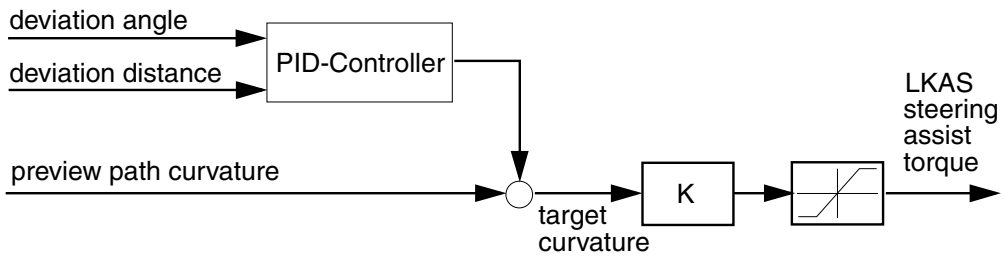


Figure 6.4: LKAS control strategy

6.6.2 Lane Departure Warning (LDW)

The *Lane Departure Warning (LDW)* system has the task to warn the driver via a signal (visual, acoustic or haptic warning signal) if the vehicle leaves the lane unintentionally.

The simple generic model uses the DLC-approch checking the *Distance to Line Crossing*. In this approach the system generates a warning, if the lateral distance of the tire to the detected line falls below a predefined threshold (see Figure 6.5).

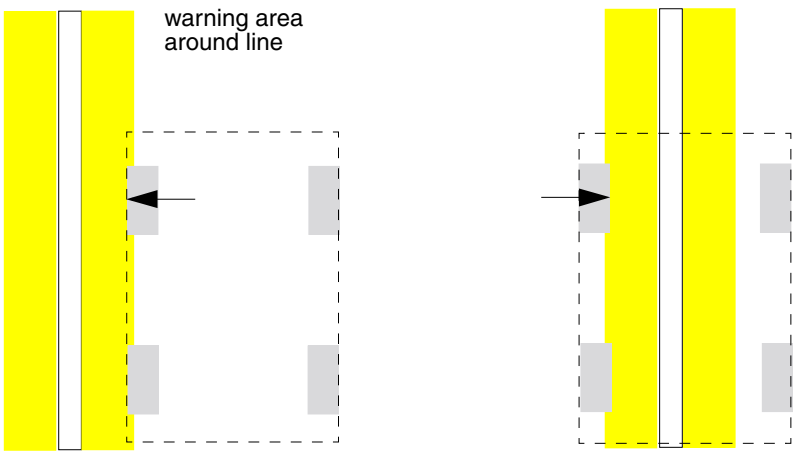


Figure 6.5: LDW active warning during line crossing

For the identification of the lines, this system needs as line detection mode the *Line Sensor*.

6.6.3 Model Parametrization

General Parameters for Generic Lateral Control

LatCtrl.LineDetectMode = *string*

Specifies the initial line detection mode, which can be modified during the simulation using DVA on the quantity `LatCtrl.LineDetectMode`. Following options are supported:

- *LineSensor*: detection using a line sensor (default)
- *RoadSensor*: detection using a road sensor

LatCtrl.RefLineSensorName = *name*

Specifies the name of the reference LineSensor, which is used by the model.
Default: *LN00*.

LatCtrl.RefRoadSensorName = *name*

Specifies the name of the reference RoadSensor, which is used by the model.
Default: *RP00*.

Parameters for LKAS

LKAS.SwitchedOn = *bool*

Switches the functionality of *Lane Keeping Assist System* on or off.
Default: 1 (=on).

LKAS.Vel_min = *value*

Specifies the minumum vehicle velocity for the activation of the lane keeping assistant.
Default: 55km/h.

LKAS.AssistTrq_max = *value*

Specifies the maximum allowed assist torque. Default: 2Nm.

LKAS.t_filter_AssistTrq = *value*

Specifies the PT1-filter time constant to filter the raw assistance torque . Default: 0.003s.

LKAS.LaneWidth_max = *width_max*
LKAS.LaneWidth_min = *width_min*

Specifies the maximum/minimum allowed lane width for the activation of the lane keeping assistant. Default: width_max = 7m, width_min=max(1.8, vehicle width + 0.1).

LKAS.kp = *P-value*
LKAS.ki = *I-value*
LKAS.kd = *D-value*

Specifies the PID-parameters of the controller for target curvature. Default: P=2, I=0.2, D=0.

LKAS.ey_int_max = *value*

Limit the integrated deviation distance value of the controller for target curvature.
Default: 10m.

LKAS.ay2Trq = *value*

Specifies the coefficient for the calculation of the assistance torque depending on lateral acceleration. Default: 2.05 Ns^2.

Parameters for LDW

LDW.SwitchedOn = *bool*

Switches the functionality of *Lane Departure Warning* on or off. Default: 1 (=on).

LDW.Vel_min = *value*

Specifies the minumum vehicle velocity for the activation of the lane departure warning.
Default: 55km/h.

LDW.dist_warn = *value*

Specifies the distance threshold for crossing lines to activate the lane departure warning.
Default: 0.2m.

6.6.4 User Accessible Quantities

Please refer to section 24.6.4 'Generic Lateral Control' on page 753.

Chapter 7

Vehicle Body

7.1 Overview

The simulated vehicle is a multi body system which is characterized through different bodies. They are generated and optimized with MESA VERDE.

Description of the bodies:

Body	Parts of the body
vehicle's body	All sprung masses beside engine, trimloads and vehicle loads.
trimloads	Constant loads to be added up to vehicle's curb load.
vehicle loads	Additional loads to define a certain load case, e.g. measuring equipment, luggage... (changeable from GUI).
wheel suspension - front left - front right - rear left - rear right	All unsprung masses without the wheel, like link, wheel carrier, suspension leg, wishbone mount...
wheel - front left - front right - rear left - rear right	All rotating masses, like tire, rim, bearing, brake drum/disc...

External and internal forces/torques and constraints are determined by:

- Suspension Force Elements
- Aerodynamics
- Kinematics and Compliance
- Tire forces/torques

- User defined virtual forces and torques

Additionally the vehicle model supports the feature to calculate *body fixed sensors* for acceleration, velocity, rotational acceleration and rotational velocity.

7.1.1 Configuration of the vehicle model

To obtain the typical behavior of a certain type of vehicle the multi body system is extensively parameterizable. The parameters are selected in a way that all type of vehicles, from a small compact car up to a big SUV can be simulated by only changing the set of parameters. There is *no* need to change the structure of the multi body system (the underlying equations).

7.1.2 Finding the equilibrium state

By pressing the start button and after initialization the vehicle starts at steady state. This is called the *start-off configuration*. This means the vehicle is in equilibrium state respecting all internal and external forces/torques (no acceleration, but non-zero velocities, in general).

Parameters for the vehicle configuration are given in the *design configuration*. The expression design configuration does not only refer to geometric quantities, but includes all other design parameters (masses, spring-stiffnesses, ...) as specified by the car manufacturer. *A design configuration usually is not a configuration of static equilibrium!*

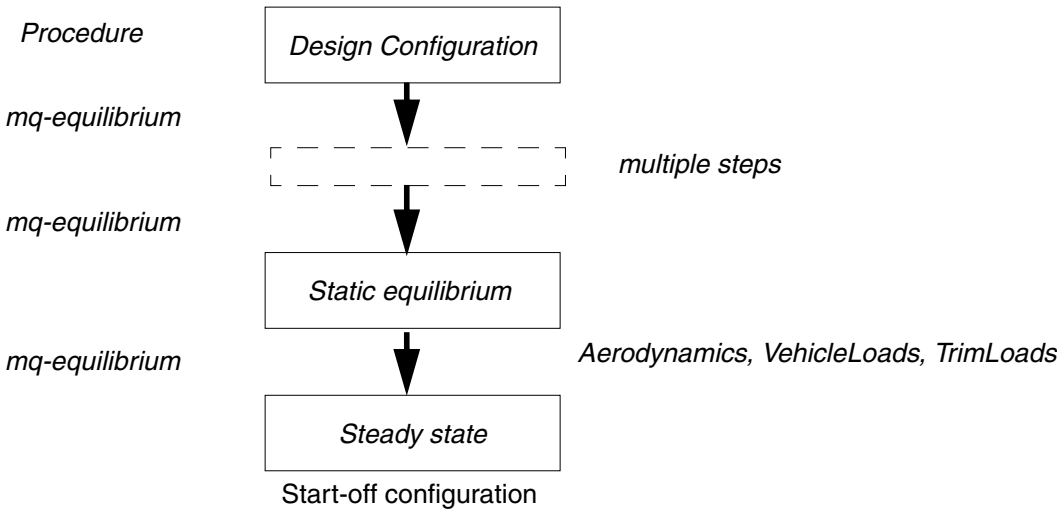


Figure 7.1: Finding steady state for the vehicle model

For computation *coordinates* q and *parameters* p are distinguished. Parameters are time invariant, coordinates are not. Parameters need not to be geometrical, but also include masses, stiffnesses, etc.

The d*esign configuration* has the coordinates and parameters (q_d, p_d). To find the steady state position of the vehicle a procedure called *Modify-q-equilibrium (mq-equilibrium)* is used.

Mq-equilibrium modifies some of the coordinates *q* such that:

- Parameters *p* obtain prescribed values.

- The configuration is an equilibrium configuration. However, at this stage no additional charges (trimloads, vehicle loads) are added to the system.

$$(q_d, p_d) \underset{\text{modify } q}{\overset{p = p_0}{\Rightarrow}} (q_e, p_e) \qquad \textbf{(EQ 9)}$$

The *start-off configuration* is a configuration that matches the initial conditions of a particular test-run. It differs from a nominal configuration (static equilibrium) by taking into account:

- Trim-loads, vehicle loads
- Start-off (or initial) driving velocity
- Aerodynamics.

The start-off configuration is obtained from the nominal configuration by modifying q , keeping p fixed:

$$(q_e, p_e) \underset{\text{modify } q}{\overset{\text{keep } p \text{ const}}{\Rightarrow}} (q_s, p_s)\,. \qquad \textbf{(EQ 10)}$$

7.1.3 Vehicle Interface

The vehicle body is the central model. It consists of the Mesa Verde multibody vehicle model along with predefined interfaces to other modules. The vehicle body interface is defined in `Vehicle.h`. Alternative vehicle models have to fill this interface with life.

7.1.4 Interaction with other modules

The vehicle body module interferes with other modules from the vehicle library.

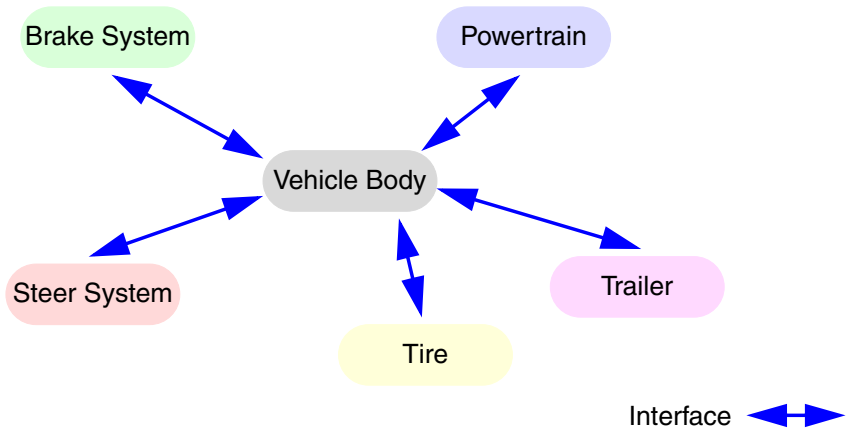


Figure 7.2: Modules interfering with the vehicle body

Each module has an interface to the vehicle body module. By these interfaces parameters and coordinates of the vehicle body are modified.

7.2 General Parameters

RefPointInputSystem = ***x*** ***y*** ***z***

This parameter specifies the origin of the `Fr1` (in `FrD` coordinates). The coordinates XYZ point from the origin of `FrD` to the origin of `Fr1`. (see section 1.2 'CarMaker Axis Systems').

Example `RefPointInputSystem = -2.0 0.0 0.0`

This means that the origin of `Fr1` is 2 m behind (in vehicles longitudinal direction) `FrD`.

Hitch.pos = ***x*** ***y*** ***z***

Hitch (trailer coupling device) position at the vehicle expressed in `FrD` coordinates.

Example `Hitch.pos = 0.0 0.0 0.3`

Virtual.PoA = ***x*** ***y*** ***z***
VirtualB.PoA = ***x*** ***y*** ***z***

The virtual force attacks the vehicle body in `Virtual.PoA`.
`Virtual.PoA` is decomposed in FrD.
Default position is the center of mass of vehicle body (see page 84).
The virtual torques always attack at center of mass of vehicle body.

VirtualB only in case of vehicle model CarFlex.

Sensor.Pol.pos = ***x_FrD*** ***y_FrC*** ***z_FrD***

The maneuver control uses this sensor signals to control the vehicle (Point of Interest). Default sensor position is the arithmetic mean value of all wheel carrier position vectors (the *z_FrD* component is multiplied by two).

Picture.PicFName = ***FName***

File name of the Tcl/Tk picture you see in the main HIL application. The file is searched in the *`Data/Pic`* folder. Unimportant for simulation results.

Example `Picture.PicFName = VW_NewBeetle.tcl`

Movie.Skin.FName = ***FName***

The file name of the picture of the car you see in IPGMovie.

Example `Movie.Skin.FName = VW_NewBeetle.obj`

Vehicle.OuterSkin = ***rll.x rll.y rll.z fur.x fur.y fur.z***

Vehicles outer skin box, defined by the points rear lower left (rll) and front upper right (fur).

Example `Vehicle.OuterSkin =-0.1 0.85 0.2  4.05 -0.85 1.6`

Instruments.vMax = ***value***

Optional. Overwrites the internally computed maximum possible vehicle velocity [km/h] for the instruments.

Instruments.Engine_rotv_Max = ***value***

Optional. Overwrites the internally computed maximum engine speed [rpm] for the instruments.

Jack.fl.pos = ***x y z***
Jack.fr.pos = ***x y z***
Jack.rl.pos = ***x y z***
Jack.rr.pos = ***x y z***

Optional. Jack positions at vehicle chassis.

nAxle = ***NumberOfAxles***

Number of vehicle axles. Possible values: 2 (3 or 4 for TruckMaker vehicle).

7.3 Mass Geometry

7.3.1 Overview Mass Geometry

The representation of the distribution of mass in a material system is called mass geometry. Inertia properties are associated with the vehicle-body, each of the four wheel carriers and each wheels. Additional body loads are modeled via additional masses and inertias as well. Three parameters have to be specified to define a mass element:

- The mass value of the body to define.
- The *center of mass* (`CoM`) of each body is defined. The center of mass is a geometrical point. The three scalar quantities are the position vector of `CoM` decomposed relative to the origin of the definition frame.
- The *inertia tensor* (= *second moments of mass = moments of inertia*) is a symmetric second-order tensor, which is specified by six scalar quantities `I = A B C D E F` (Frequently the off-diagonal elements (`D, E, F`) of the inertia tensor are neglected):

$$\begin{bmatrix} A & F & E \\ F & B & D \\ E & D & C \end{bmatrix} = \begin{bmatrix} \int_{\text{Body}} (y^2+z^2)dm & -\int_{\text{Body}} xydm & -\int_{\text{Body}} xzdm \\ \text{symmetry} & \int_{\text{Body}} (z^2+x^2)dm & -\int_{\text{Body}} yzdm \\ \text{symmetry} & \text{symmetry} & \int_{\text{Body}} (x^2+y^2)dm \end{bmatrix} \quad \textbf{(EQ 11)}$$

7.3.2 Parameters

Parameters The following parameters are required for this model:

Body.pos
Body.mass
Body.I

The vehicle body without engine. Its center of mass is placed in the design frame at *Body.Pos*. The vehicle body has the mass *Body.mass* and the inertia tensor *Body.I*. The elements of the inertia tensor are given in the order A B C D E F. It is sufficient to give the elements A B C (main diagonal elements) only.

A vehicle body with inertia A=360 kg*m^2, B=800 kg*m^2, C=1800 kg*m^2:

```
Body.mass =     375.0
Body.Pos =      1.5  0.0  0.45
Body.I =        360  800  1800
```

WheelCarrier.fl.pos
WheelCarrier.fl.mass
WheelCarrier.fl.I

Wheel carrier front left. Represents all unsprung mass in the suspension (except the wheel).

WheelCarrier.fr.pos
WheelCarrier.fr.mass
WheelCarrier.fl.I

Wheel carrier front right. For details, see wheel carrier front left.

WheelCarrier.rl.pos
WheelCarrier.rl.mass
WheelCarrier.rl.I

Wheel carrier rear left. For details, see wheel carrier front left.

WheelCarrier.rr.pos
WheelCarrier.rr.mass
WheelCarrier.rr.I

Wheel carrier rear right. For details, see wheel carrier front left.

Wheel.fl.pos
Wheel.fl.mass
Wheel.fl.I

Wheel front left. The wheel and all other rotating components (parts of the brake, ...) .

Wheel.fr.pos
Wheel.fr.mass
Wheel.fl.I

Wheel front right. For details, see wheel front left.

Wheel.rl.pos
Wheel.rl.mass
Wheel.rl.I

Wheel rear left. For details, see wheel front left.

Wheel.rr.pos
Wheel.rr.mass
Wheel.rr.I

Wheel rear right. For details, see wheel front left.

TrimLoad.<i>.pos = ***x y z***
TrimLoad.<i>.mass = ***mass_kg***
TrimLoad.<i>.I = ***Ixx Iyy Izz ...***

Bodies to trim mass contribution to that of a reference vehicle. Trim loads are fixed to vehicle body. <i> := 0, 1, 2, ...

Do not confuse TrimLoads with test run specific additional charges you may want to put on your vehicle.

If *mass_kg* is zero, the load is ignored.

TrimLoad.<i>.mounted = *MountFrame*

Indicates the frame on which the trimload is mounted. Following frames are possible: `Fr1A` and `Fr1B`. Default: `Fr1A`.

7.3.3 Vehicle with Twin Tires

Generally the vehicle uses single tires for each wheel. In this case, the wheel carrier and the corresponding tire must be defined in the same position.

AxleR.TwinTiresOn = *bool*

Alternativly the vehicle can be modeled with a rear axle using twin tires. This parameter is used to activate twin tires for the rear axle. Default: 0.

In this case, the wheel carrier is supposed to be in the middle between the two corresponding tires. The y-Position of the wheel carrier and the corresponding outer wheel in the GUI can diversify. The y-position of the second wheel (inner wheel) is mirror-inverted relating to the wheel carrier.

Using twin tires there are also two contact points with tire forces and moments. Additional User Accessible Quantities are defined for the second (inner) tire. These quantities uses a prefix ‘Twin’.

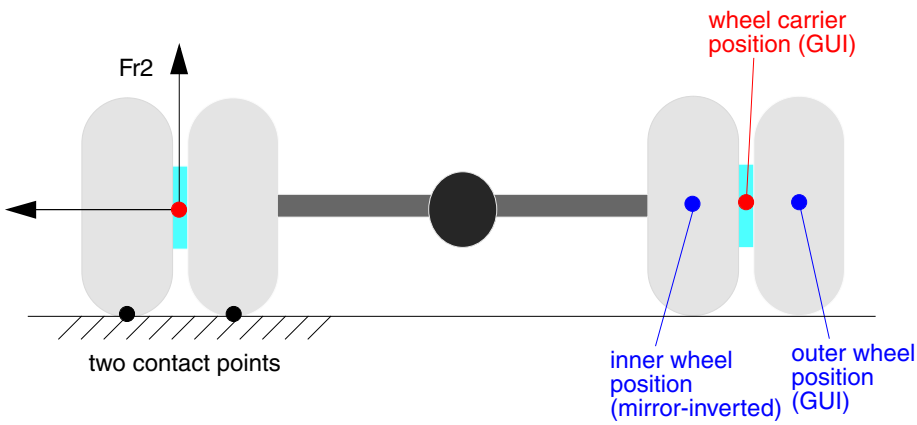


Figure 7.3: Schema of a twin wheel

7.4 User Accessible Quantities for Vehicle Body

Please refer to section 24.5 'Vehicle'.

Chapter 8

Vehicle Body Modelling

8.1 Equations

Like mentioned in the chapter before, the vehicle body is a multi body system which is characterized through different bodies. The motion of the multi body system is described with differential and with algebraic equations.

Differential Equations (Motion equations)

The *principle of d'Alembert* is applied to get differential equations of motion for the generalized coordinates of the system. These are obtained in the following form

$$A\ddot{q} = B(\dot{q}, q) \qquad \text{(EQ 12)}$$

A is the mass matrix, the vector $\ddot{q}$ contains the second order derivatives of the generalized coordinates of the system, the matrix B contains the projection of centrifugal, gyroscopic, and applied forces onto the free modes of motion. The vehicle body motion is decribed by following generalized coordinates:

Generalized coordinate	Unit	Info
q_{tx}, q_{ty}, q_{tz}	m	x,y,z-position of vehicle body frame Fr_1 relative to global frame Fr_0, expressed in global frame Fr_0
q_{rx}, q_{ry}, q_{rz}	rad	x,y,z-rotation of vehicle body frame Fr_1 relative to global frame Fr_0 with rotation order Z-Y-X
q_{0ij} with i=F,R; j=L,R	m	wheel compression
q_{rxA2B}, q_{ryA2B}	rad	x,y-rotation of vehicle body frame Fr_B relative to vehicle body Fr_A (=Fr_1) with rotation order X-Y (only with flexible vehicle body)

These motion equations are symbolically derived in a well-established formal way with the help of the MESA VERDE programm.

Additionally to the generalized coordinates q_i, there exist further degrees of freedom, which are not calculated by MESA VERDE. These are the steering rack position q_{Steer} and the angular velocity ω_{ij} of wheel relative to wheel carrier.

Algebraic Equations

The algebraic equations allow for the calculation of $B(\dot{q}, q)$, e.g. gyroscopic moments, spring/damper forces, etc. These equations are not calculated by MESA VERDE.

8.2 Model description

8.2.1 Architecture of forces

Figure 8.1 shows the internal and external forces/torques which are involved in the vehicle model calculation. The steering rack influences the elastokinematics and the kinematic constraints.

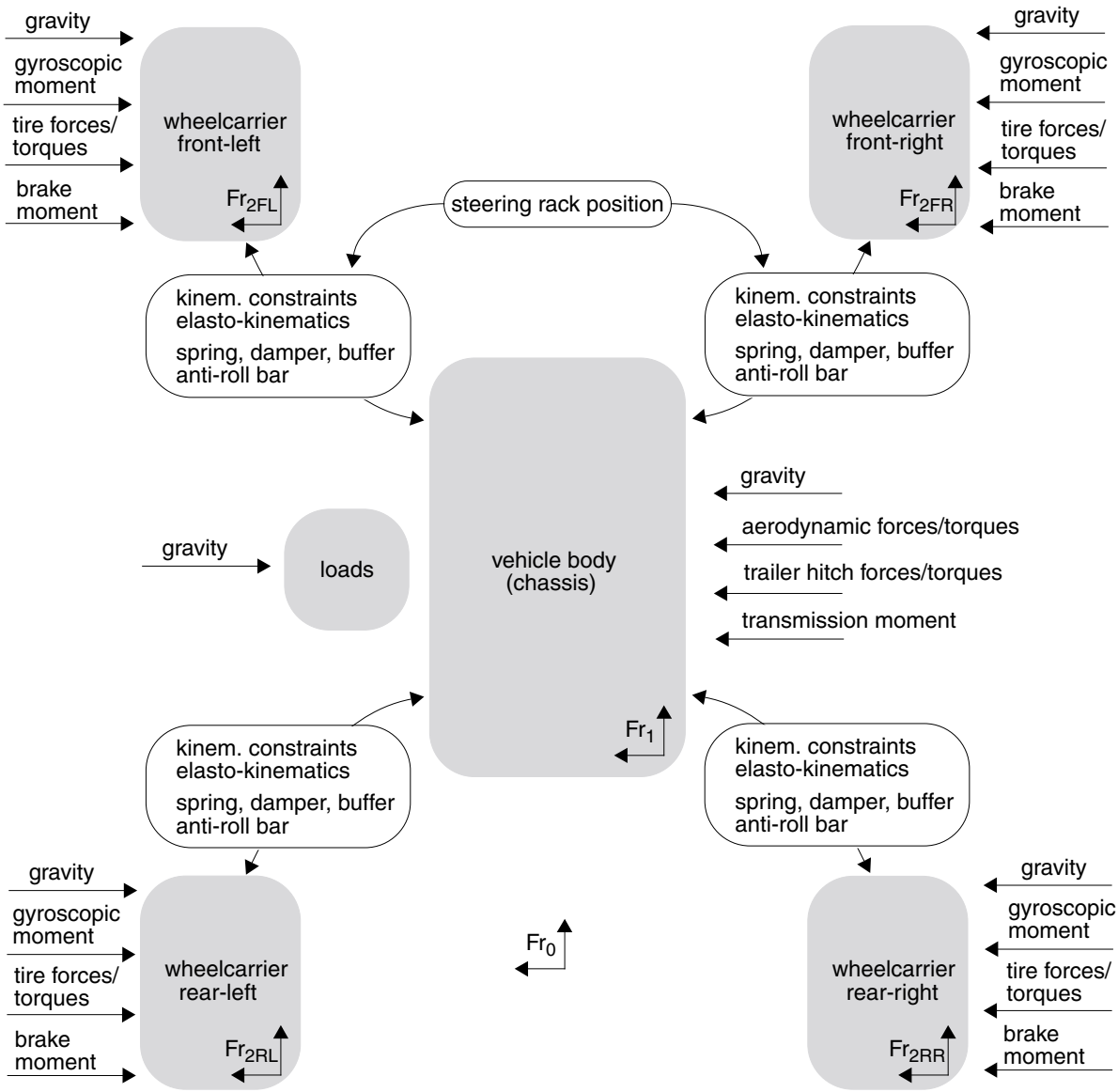


Figure 8.1: Forces involved in the vehicle model

8.2.2 Chain of calculation

Figure 8.2 demostrates the chain of calculation of the vehicle model. This corresponds to the function *Vhcl_Calc()* in *src/CM_Vehicle.c*, which is called once each simulation cycle.

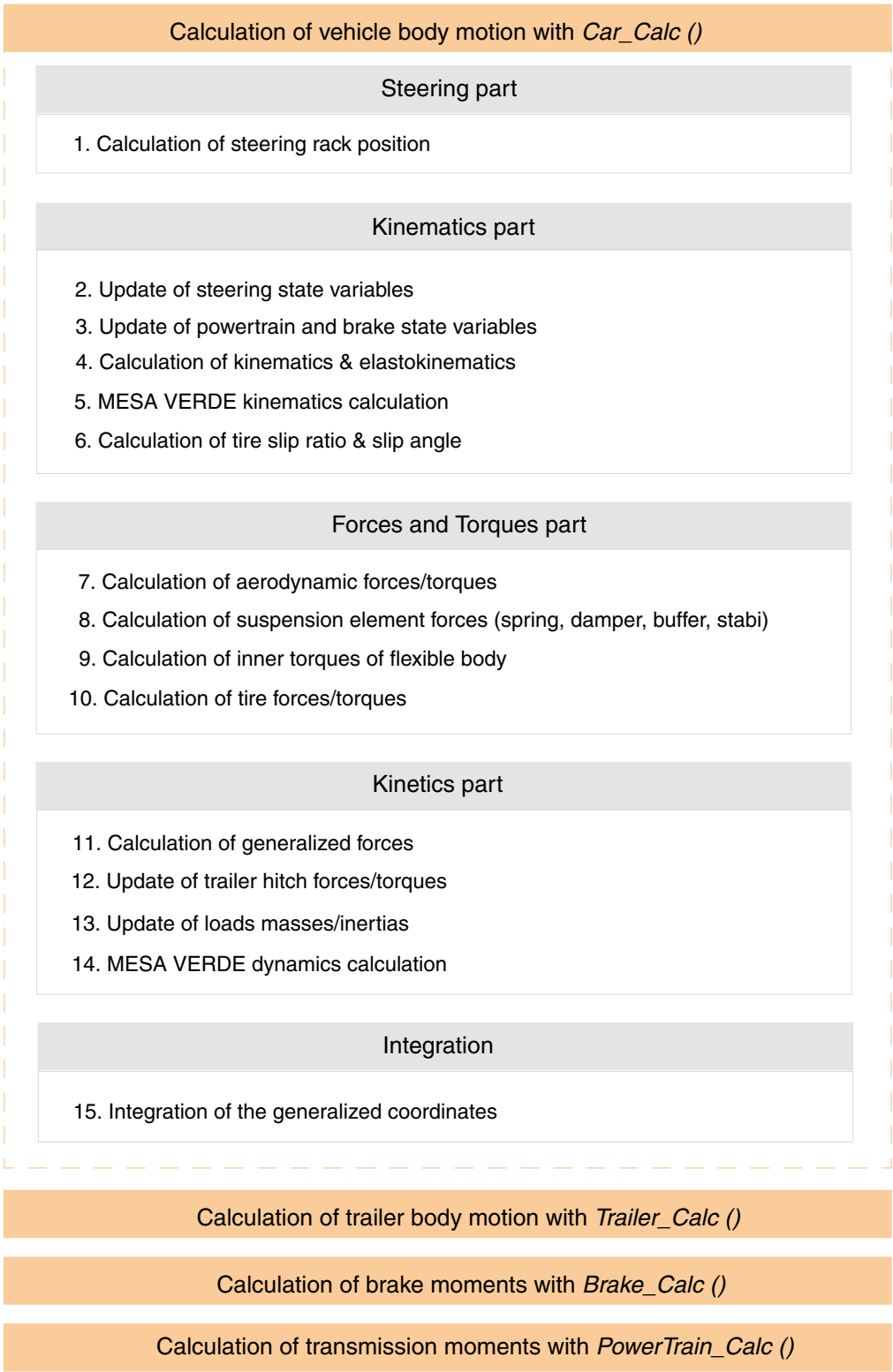


Figure 8.2: Chain of calculation

8.2.3 Description of the calculation parts

In this chapter all calculation parts of the function *Car_Calc ()* are described in detail.

Part 1: Calculation of steering rack position

In this function part the degree of freedom steering rack position q_{Steer} is calculated.

If the steering model 'steer by angle' is used, the steering rack position depends directly on the driver steering wheel angle.

If the steering model 'steer by torque' is used, the steering rack position is described by a differential equation (for details see section 14.1):

$$\ddot{q}_{Steer} \cdot m = Q_{SteerFL} + Q_{SteerFR} + F_{Driver} \quad \textbf{(EQ 13)}$$

$Q_{SteerFL}$ and $Q_{SteerFR}$ represent the generalized left and right rack forces. These forces are calculated in calculation part 11.

Part 2: Update of steering state variables

In this function part the internal steering model state variable of steering rack position q_{Steer} is copied in the vehicle model environment.

Part 3: Update of powertrain and brake state variables

In this function part the internal powertrain model and brake model state variables are copied in the vehicle environment. These variables, calculated in the functions *Brake_Calc* and *PowerTrain_Calc* are:

State variable	Unit	Info
ω_{ij}	rad/s	angular velocity of the wheel carrier *ij* with respect to the chassis
$\left[\vec{T}(O_{2ij})^{brake}\right]_{2ij}$	Nm	brake moment support on wheel carrier *ij*, expressed in wheel carrier frame Fr_{2ij}
$\left[\vec{T}(O_1)^{transm}\right]_1$	Nm	transmission moment support on chassis body A and B, expressed in chassis frame Fr_1

Part 4: Calculation of kinematics & elastokinematics

Figure 8.3 represents the multibody system which consists of five rigid bodies interconnected by five joints. The rigid bodies may move in space and relative to each body. The chassis body (Fr_1) is connected to the ground body (Fr_0) with a 6 degree of freedom joint. The wheel carrier bodies (Fr_{2ij}) are connected with the chassis body with a complex 1 degree of freedom joint. Between the front (rear) left and front (rear) right wheel carrier a kinematical dependance (coupler) can be modelled, e.g. a rigid axle.

The wheel carrier center follows a trajectory through 3D space, relative to the chassis, as function of the suspension compressions and the steering rack position.

The relative position $\vec{t}_{ij}$ and orientation $\vec{r}_{ij}$ of the wheel carrier relative to the chassis has a kinematical and an elastokinematical part:

$$\vec{t}_{ij} = \vec{t}_{ij}^{kin} + \vec{t}_{ij}^{com} \quad \textbf{(EQ 14)}$$

$$\vec{r}_{ij} = \vec{r}_{ij}^{kin} + \vec{r}_{ij}^{com} \quad \textbf{(EQ 15)}$$

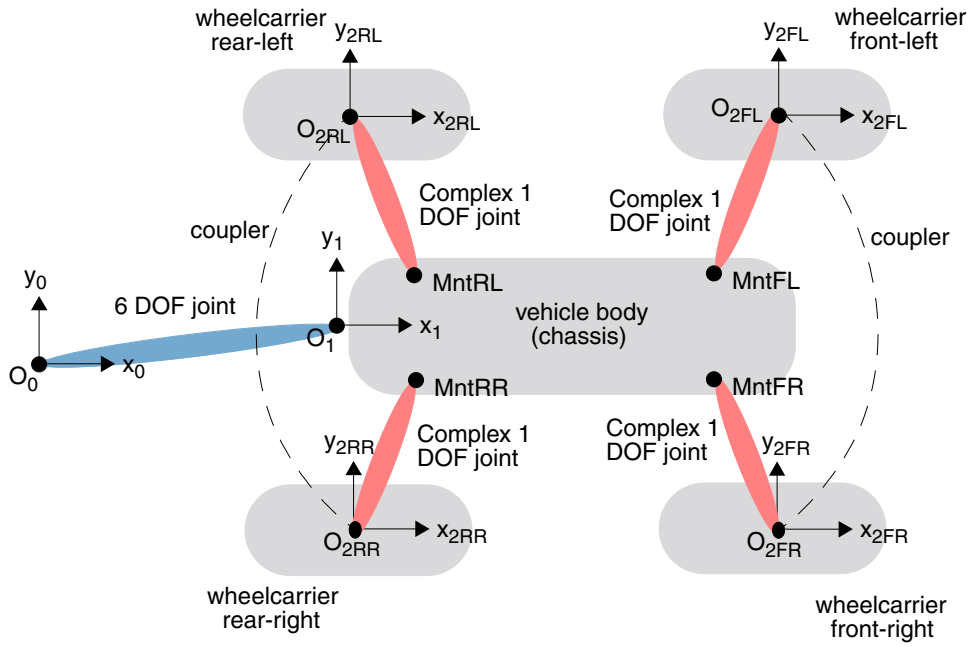


Figure 8.3: Joints within the vehicle module

Calculation of kinematics

The kinematical relative position of the wheel carrier with respect to the chassis is dependent on the suspension compressions, both of the left and right side q_{0iL}, q_{0iR}, and for the front axle on the steering rack position q_{Steer}:

$$\vec{t}_{ij}^{kin} = \overrightarrow{Mnt_{ij}O_{2ij}} = \begin{bmatrix} t_{ij,x}^{kin} \\ t_{ij,y}^{kin} \\ t_{ij,z}^{kin} \end{bmatrix}_1 = \begin{bmatrix} t_{ij,x}^{kin}(q_{0iL}, q_{0iR}, q_{Steer}) \\ t_{ij,y}^{kin}(q_{0iL}, q_{0iR}, q_{Steer}) \\ t_{ij,z}^{kin}(q_{0iL}, q_{0iR}, q_{Steer}) \end{bmatrix}_1 \quad \text{(EQ 16)}$$

The relative orientation $\vec{r}_{ij}$ of the wheel carrier with respect to the chassis contains three skalar Cardan angles to describe the orientation using the Z-X-Y rotating sequence. The kinematical relative orientation is also dependent on the suspension compressions and for the front axle on the steering rack position:

$$\vec{r}_{ij}^{kin} = \begin{bmatrix} r_{ij,x}^{kin} \\ r_{ij,y}^{kin} \\ r_{ij,z}^{kin} \end{bmatrix} = \begin{bmatrix} r_{ij,x}^{kin}(q_{0iL}, q_{0iR}, q_{Steer}) \\ r_{ij,y}^{kin}(q_{0iL}, q_{0iR}, q_{Steer}) \\ r_{ij,z}^{kin}(q_{0iL}, q_{0iR}, q_{Steer}) \end{bmatrix} \quad \text{(EQ 17)}$$

Additionally to the relative position and orientation, also the length of the suspension force elements (spring, damper, buffer, anti-roll stabilizer) depends on the suspension compressions and the steering rack position:

$$\begin{bmatrix} l_{ij}^{Spring} \\ l_{ij}^{Damper} \\ l_{ij}^{Buffer} \\ l_{ij}^{Stabi} \end{bmatrix} = \begin{bmatrix} l_{ij}^{Spring}(q_{0iL}, q_{0iR}, q_{Steer}) \\ l_{ij}^{Damper}(q_{0iL}, q_{0iR}, q_{Steer}) \\ l_{ij}^{Buffer}(q_{0iL}, q_{0iR}, q_{Steer}) \\ l_{ij}^{Stabi}(q_{0iL}, q_{0iR}, q_{Steer}) \end{bmatrix} \quad \text{(EQ 18)}$$

The dependence on the suspension compressions and the steering rack position is subjected to linear or nonlinear constraints.

In the case of linear constraints, the relative position, orientation and suspension force element lenghs are calculated with constant partial derivatives:

$$t_{ij,y}^{kin} = \frac{\partial t_{ij,y}^{kin}}{\partial q_{0iL}} \cdot q_{0iL} + \frac{\partial t_{ij,y}^{kin}}{\partial q_{0iR}} \cdot q_{0iR} + \frac{\partial t_{ij,y}^{kin}}{\partial q_{Steer}} \cdot q_{Steer} = c_0 \cdot q_{0iL} + c_1 \cdot q_{0iR} + c_2 \cdot q_{Steer} \quad \textbf{(EQ 19)}$$

$$r_{ij,y}^{kin} = \frac{\partial r_{ij,y}^{kin}}{\partial q_{0iL}} \cdot q_{0iL} + \frac{\partial r_{ij,y}^{kin}}{\partial q_{0iR}} \cdot q_{0iR} + \frac{\partial r_{ij,y}^{kin}}{\partial q_{Steer}} \cdot q_{Steer} = c_0 \cdot q_{0iL} + c_1 \cdot q_{0iR} + c_2 \cdot q_{Steer} \quad \textbf{(EQ 20)}$$

$$l_{ij}^{Spring} = \frac{\partial l_{ij}^{Spring}}{\partial q_{0iL}} \cdot q_{0iL} + \frac{\partial l_{ij}^{Spring}}{\partial q_{0iR}} \cdot q_{0iR} + \frac{\partial l_{ij}^{Spring}}{\partial q_{Steer}} \cdot q_{Steer} = c_0 \cdot q_{0iL} + c_1 \cdot q_{0iR} + c_2 \cdot q_{Steer} \quad \textbf{(EQ 21)}$$

In the case of nonlinear constraints, the relative position, orientation and suspension force element lengths are deposited in nonlinear mappings:

$$t_{ij,y}^{kin}, r_{ij,y}^{kin}, l_{ij}^{Spring} = f(q_{0iL}, q_{0iR}, q_{Steer}) \quad \textbf{(EQ 22)}$$

Because the direction of the wheel trajectory determines how the tire forces are transmitted to the chassis body, the partial derivatives are established also in case of nonlinear mappings for further calculations in the working point, see Figure 8.4.

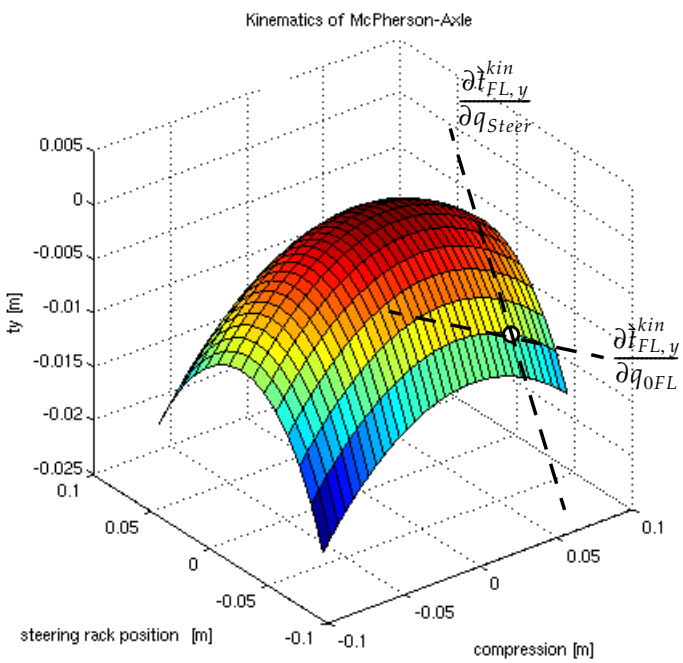


Figure 8.4: Nonlinear mapping for $t_{FL,y}^{kin} = f(q_{0FL}, q_{Steer})$

In the same manner the other relative positions $t_{ij,x}^{kin}$, $t_{ij,z}^{kin}$ such as other relative orientations $r_{ij,x}^{kin}$, $r_{ij,z}^{kin}$ and other suspension force element lengths l_{ij}^{Damper}, l_{ij}^{Buffer}, l_{ij}^{Stabi} are calculated.

Calculation of elastokinematics

In practice, suspension system joints are typically mounted with elastic (rubber) bushings. These bushings allow small motions agains the “kinematically” desired motion of each joint. These small motions express the deformation due to exciting forces and torques in the wheel carrier:

$$t_{ij,x}^{com} = \left[\frac{\partial t_{ij,x}^{com}}{\partial F_{iL,k}}\right] \cdot \vec{F}_{iL}^{WC} + \left[\frac{\partial t_{ij,x}^{com}}{\partial F_{iR,k}}\right] \cdot \vec{F}_{iR}^{WC} + \left[\frac{\partial t_{ij,x}^{com}}{\partial T_{iL,k}}\right] \cdot \vec{T}(O_{2iL})^{WC} + \left[\frac{\partial t_{ij,x}^{com}}{\partial T_{iR,k}}\right] \cdot \vec{T}(O_{2iR})^{WC} \quad \textbf{(EQ 23)}$$

$$r_{ij,x}^{com} = \left[\frac{\partial r_{ij,x}^{com}}{\partial F_{iL,k}}\right] \cdot \vec{F}_{iL}^{WC} + \left[\frac{\partial r_{ij,x}^{com}}{\partial F_{iR,k}}\right] \cdot \vec{F}_{iR}^{WC} + \left[\frac{\partial r_{ij,x}^{com}}{\partial T_{iL,k}}\right] \cdot \vec{T}(O_{2iL})^{WC} + \left[\frac{\partial r_{ij,x}^{com}}{\partial T_{iR,k}}\right] \cdot \vec{T}(O_{2iR})^{WC} \quad \textbf{(EQ 24)}$$

Variable	Unit	Info
$\left[\vec{F}_{ij}^{WC}\right]_1 = \begin{bmatrix} F_{ij,x}^{WC} \\ F_{ij,y}^{WC} \\ F_{ij,z}^{WC} \end{bmatrix}_1$	N	total applied forces of the wheel carrier *ij (i=F,R; j=L,R)* onto the chassis, expressed in chassis frame Fr_1
$\left[\vec{T}(O_{2ij})^{WC}\right]_1 = \begin{bmatrix} T(O_{2ij})_x^{WC} \\ T(O_{2ij})_y^{WC} \\ T(O_{2ij})_z^{WC} \end{bmatrix}_1$	Nm	total applied torques of the wheel carrier *ij (i=F,R; j=L,R)* onto the chassis in the wheel carrier center O_{2ij}, expressed in chassis frame Fr_1
$\left[\frac{\partial t_{ij,x}^{com}}{\partial F_{iL,k}}\right]_1 = \begin{bmatrix} \frac{\partial t_{ij,x}^{com}}{\partial F_{iL,x}} \\ \frac{\partial t_{ij,x}^{com}}{\partial F_{iL,y}} \\ \frac{\partial t_{ij,x}^{com}}{\partial F_{iL,z}} \end{bmatrix}_1, \left[\frac{\partial t_{ij,x}^{com}}{\partial F_{iR,k}}\right]_1$	m/N	partial derivation of the elastokinematical relative x-position between the wheel carrier and the chassis with respect to exciting force $F_{k=x,y,z}$ in the wheel carrier *iL* and *iR*
$\left[\frac{\partial r_{ij,x}^{com}}{\partial F_{iL,k}}\right]_1 = \begin{bmatrix} \frac{\partial r_{ij,x}^{com}}{\partial F_{iL,x}} \\ \frac{\partial r_{ij,x}^{com}}{\partial F_{iL,y}} \\ \frac{\partial r_{ij,x}^{com}}{\partial F_{iL,z}} \end{bmatrix}_1, \left[\frac{\partial r_{ij,x}^{com}}{\partial F_{iR,k}}\right]_1$	rad/N	partial derivation of the elastokinematical relative x-rotation between the wheel carrier and the chassis with respect to exciting force $F_{k=x,y,z}$ in the wheel carrier *iL* and *iR*

The partial derivations $\frac{\partial t_{ij,x}^{com}}{\partial F_{ij,k}}$, $\frac{\partial t_{ij,x}^{com}}{\partial T_{ij,k}}$, $\frac{\partial r_{ij,x}^{com}}{\partial F_{ij,k}}$ and $\frac{\partial r_{ij,x}^{com}}{\partial T_{ij,k}}$ are defined constant parameters in the vehicle data set.

In the same manner the other directions $t_{ij,y}^{com}$, $t_{ij,z}^{com}$ such as $r_{ij,y}^{com}$, $r_{ij,z}^{com}$ are calculated.

For details see section 12.1 et seq.

Part 5: MESA VERDE kinematics calculation

In this function part the first set of MESA VERDE equations is called for the kinematics calculation. This part calculates following variables:

Variable	Unit	Info
$\left[\overrightarrow{O_0 O_{2ij}}\right]_0$	m	global position of the wheel carrier *ij*, expressed in global frame Fr_0
$\left[\vec{V}_{(2ij)/0}(O_{2ij})\right]_0$	m/s	global velocity of the wheel carrier *ij* in the wheel center O_{2ij}, expressed in global frame Fr_0
$\left[\vec{\Omega}_{(2ij)/0}\right]_0$	rad/s	global angular velocity of the wheel carrier *ij*, expressed in global frame Fr_0
$R_{02ij} = R_{2ij \to 0}$		transformation matrix from wheel carrier frame Fr_{2ij} into global frame Fr_0

Part 6: Calculation of tire slip ratio & slip angle

This function part calculates the tire slip ratio s_{ij} and slip angle α_{ij} of each tire as function of following variables:

$$s_{ij}, \alpha_{ij} = f(\vec{V}_{(2ij)/0}(O_{2ij}), \omega_{ij}) \quad \textbf{(EQ 25)}$$

Variable	Unit	Info
s_{ij}		longitudinal slip ratio of tire *ij*
α_{ij}	rad	sideslip angle of tire *ij*

For details, see section 17.1.1.

Part 7: Calculation of aerodynamic forces/torques

This function part calculates the aerodynamic forces $\vec{F}^{aero}$ and torques $\vec{T}(O_A)^{aero}$ depending on the global vehicle velocitiy and the wind velocity:

$$\vec{F}^{aero}, \vec{T}(O_A)^{aero} = f(\vec{V}_{1/0}(O_A), \vec{V}_{Wind/0}) \quad \textbf{(EQ 26)}$$

Variable	Unit	Info
$\left[\vec{F}^{aero}\right]_1 = \begin{bmatrix} F_x^{aero} \\ F_y^{aero} \\ F_z^{aero} \end{bmatrix}_1$	N	aerodynamic forces, expressed in chassis frame Fr_1
$\left[\vec{T}(O_A)^{aero}\right]_1 = \begin{bmatrix} T(O_A)_x^{aero} \\ T(O_A)_y^{aero} \\ T(O_A)_z^{aero} \end{bmatrix}_1$	Nm	aerodynamic torques acting in the aerodynamic frame origin O_A, expressed in chassis frame Fr_1
$\left[\vec{V}_{1/0}(O_A)\right]_1$	m/s	global vehicle velocity in the aerodynamic frame origin O_A, expressed in chassis frame Fr_1
$\left[\vec{V}_{Wind/0}\right]_1$	m/s	global wind velocity, expressed in chassis frame Fr_1

For details, see section 13.1.

Part 8: Calculation of suspension element forces

The vehicle suspension, which are mounted between the chassis body and the wheel carrier body, contains four force elements: spring, damper, buffer and roll stabilizer.

The suspension element forces are calculated as function of the corresponding suspension element length or velocity:

$$F_{ij}^{Spring} = f(l_{ij}^{Spring}) \quad \text{(EQ 27)}$$

$$F_{ij}^{Damper} = f(l_{ij}^{Damper}) \quad \text{(EQ 28)}$$

$$F_{ij}^{Buffer} = f(l_{ij}^{Buffer}) \quad \text{(EQ 29)}$$

$$F_{ij}^{Stabi} = f(l_{iL}^{Stabi}, l_{iR}^{Stabi}) \quad \text{(EQ 30)}$$

Variable	Unit	Info
F_{ij}^{Spring}, F_{ij}^{Damper}, F_{ij}^{Buffer}, F_{ij}^{Stabi}	N	suspension element force of suspension *ij*
l_{ij}^{Spring}, l_{ij}^{Buffer}, l_{ij}^{Stabi}	m	suspension element length of suspension *ij*
$l_{ij}^{Damper} = \frac{dl_{ij}^{Damper}}{dt}$	m/s	damper velocity of suspension *ij*

For details, see section 11.2 et seq.

Part 9: Calculation of inner torques of flexible body

The chassis body can be modelled as a flexible body with two additional generalized coordinates q_{rxA2B} and q_{ryA2B}. The inner torques between the frame Fr_A and Fr_B are dependent on the two generalized coordinates and their derivatives:

$$\vec{T}(O_1)^{A2B} = f(q_{rxA2B}, \dot{q}_{rxA2B}, q_{ryA2B}, \dot{q}_{ryA2B}) \quad \text{(EQ 31)}$$

Variable	Unit	Info
$\left[\vec{T}(O_1)^{A2B}\right]_1 = \begin{bmatrix} T(O_1)_x^{A2B} \\ T(O_1)_y^{A2B} \\ 0 \end{bmatrix}_1$	Nm	Inner torques between the frame Fr_A and Fr_B in the origin O_1, expressed in chassis frame Fr_1

Part 10: Calculation of tire forces/torques

In this part the tire forces and tire torques in the wheel carrier are calculated. These forces and torques depend on the normal load, slip ratio and slip angle:

$$F_{ij,x}^{tire}, F_{ij,y}^{tire} = f(F_{ij,z}^{tire}, s_{ij}, \alpha_{ij}) \quad \text{(EQ 32)}$$

$$\vec{T}(O_{2ij})^{tire} = f(F_{ij,z}^{tire}, s_{ij}, \alpha_{ij}) \quad \text{(EQ 33)}$$

Variable	Unit	Info
$\left[\vec{F}_{ij}^{tire}\right]_{2ij} = \begin{bmatrix} F_{ij,x}^{tire} \\ F_{ij,y}^{tire} \\ F_{ij,z}^{tire} \end{bmatrix}_{2ij}$	N	tire forces in the tire *ij*, expressed in wheel carrier frame Fr_{2ij}
$\left[\vec{T}(O_{2ij})^{tire}\right]_{2ij} = \begin{bmatrix} T(O_{2ij})_x^{tire} \\ T(O_{2ij})_y^{tire} \\ T(O_{2ij})_z^{tire} \end{bmatrix}_{2ij}$	Nm	tire torques acting in the wheel carrier center O_{2ij}, expressed in wheel carrier frame Fr_{2ij}

Part 11: Calculation of generalized forces

This calculation part handles the calculation of the gyroscopic moments, the calculation of total applied forces/torques in the wheel carrier and the calculation of generalized forces.

Calculation of gyroscopic moments

The gyroscopic moment is calculated in following way:

$$\left[\vec{T}(O_{2ij})^{gyro}\right]_{2ij} = \begin{bmatrix} I_{ij,yy} \cdot \Omega_{(2ij)/0,z} \cdot \omega_{ij} \\ 0 \\ -I_{ij,yy} \cdot \Omega_{(2ij)/0,x} \cdot \omega_{ij} \end{bmatrix}_{2ij} \quad \text{(EQ 34)}$$

Variable	Unit	Info
$\left[\vec{T}(O_{2ij})^{gyro}\right]_{2ij} = \begin{bmatrix} T(O_{2ij})_x^{gyro} \\ 0 \\ T(O_{2ij})_z^{gyro} \end{bmatrix}_{2ij}$	Nm	gyroscopic moments in the wheel carrier *ij*, expressed in wheel carrier frame Fr_{2ij}
$I_{ij,yy}$	kgm^2	component along rotation axis of the wheel *ij* inertia
$\left[\vec{\Omega}_{(2ij)/0}\right]_{2ij} = \begin{bmatrix} \Omega_{(2ij)/0,x} \\ \Omega_{(2ij)/0,y} \\ \Omega_{(2ij)/0,z} \end{bmatrix}_{2ij}$	rad/s	global angular velocity of the wheel carrier *ij*, expressed in wheel carrier frame Fr_{2ij}

Calculation of the total applied wheel carrier forces and torques

For the calculation of generalized forces the total applied wheel carrier forces and torques are required. This is the sum of all acting forces and torques to the wheel carrier:

$$\left[\vec{F}_{ij}^{WC}\right]_{2ij} = \left[\vec{F}_{ij}^{tire}\right]_{2ij} + \left[\vec{F}_{ij}^{ext}\right]_{2ij} \quad \text{(EQ 35)}$$

$$\left[\vec{T}(O_{2ij})^{WC}\right]_{2ij} = \left[\vec{T}(O_{2ij})^{ext}\right]_{2ij} + \left[\vec{T}(O_{2ij})^{gyro}\right]_{2ij} + \left[\vec{T}(O_{2ij})^{tire}\right]_{2ij} + \left[\vec{T}(O_{2ij})^{brake}\right]_{2ij} \quad \text{(EQ 36)}$$

The corresponding total applied wheel carrier forces and torques, decomposed in chassis frame Fr_1, are:

$$\left[\vec{F}_{ij}^{WC}\right]_{1} = R_{2ij \to 1} \cdot \left[\vec{F}_{ij}^{WC}\right]_{2ij} \quad \text{(EQ 37)}$$

$$\left[\vec{T}(O_{2ij})^{WC}\right]_{1} = R_{2ij \to 1} \cdot \left[\vec{T}(O_{2ij})^{WC}\right]_{2ij} \quad \text{(EQ 38)}$$

Variable	Unit	Info
$\left[\vec{F}_{ij}^{WC}\right]_{2ij} = \begin{bmatrix} F_{ij,x}^{WC} \\ F_{ij,y}^{WC} \\ F_{ij,z}^{WC} \end{bmatrix}_{2ij}$, $\left[\vec{F}_{ij}^{WC}\right]_{1}$	N	total applied forces in the wheel carrier *ij*, expressed in wheel carrier frame Fr_{2ij} and in chassis frame Fr_1
$\left[\vec{F}_{ij}^{ext}\right]_{2ij}$	N	external used defined forces in the wheel carrier *ij*, expressed in wheel carrier frame Fr_{2ij}
$\left[\vec{T}(O_{2ij})^{WC}\right]_{2ij} = \begin{bmatrix} \vec{T}(O_{2ij})_x^{WC} \\ \vec{T}(O_{2ij})_y^{WC} \\ \vec{T}(O_{2ij})_z^{WC} \end{bmatrix}_{2ij}$ $\left[\vec{T}(O_{2ij})^{WC}\right]_{1}$	Nm	total applied torques in the wheel carrier center O_{2ij}, expressed in wheel carrier frame Fr_{2ij} and in chassis frame Fr_1
$\left[\vec{T}(O_{2ij})^{ext}\right]_{2ij}$	Nm	external used defined torques in the wheel carrier origin O_{2ij}, expressed in wheel carrier frame Fr_{2ij}
$R_{12ij} = R_{2ij \to 1}$		transformation matrix from wheel carrier frame Fr_{2ij} into chassis frame Fr_1

Calculation of the generalized forces for q_0 and q_{Steer}

In general, with the aid of the *Lagrange's Equations*, the generalized force Q_i associated to the generalized coordinate q_i is calculated in following way:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) - \frac{\partial T}{\partial q_i} = Q_i \text{ with } Q_i = \sum_k F_k \cdot \frac{\partial x_k}{\partial q_i}, \quad \text{(EQ 39)}$$

where T is a kinetic energy of the system.

For the generalized coordinate q_{0RL} the generalized force Q_{0RL} becomes:

$$Q_{0RL} = \left[\vec{F}_{RL}^{WC}\right]_1 \cdot \left[\frac{\partial \vec{t}_{RL}^{kin}}{\partial q_{0RL}}\right]_1 + \left[\vec{F}_{RR}^{WC}\right]_1 \cdot \left[\frac{\partial \vec{t}_{RR}^{kin}}{\partial q_{0RL}}\right]_1 + \left[\vec{T}(O_{2RL})^{WC}\right]_1 \cdot \left[\vec{\Pi}_{2RL,\, q_{0RL}}\right]_1 +$$

$$\left[\vec{T}(O_{2RR})^{WC}\right]_1 \cdot \left[\vec{\Pi}_{2RR,\, q_{0RL}}\right]_1 + \sum_X F_{RL}^{SElmX} \cdot \frac{\partial l_{RL}^{SElmX}}{\partial q_{0RL}} + \sum_X F_{RR}^{SElmX} \cdot \frac{\partial l_{RR}^{SElmX}}{\partial q_{0RL}} \qquad \textbf{(EQ 40)}$$

Variable	Unit	Info
F_{ij}^{SElmX}	N	suspension element force of suspension *ij;* x=spring, damper, buffer, anti-roll bar
$\left[\vec{\Pi}_{2Rij,\, q_{0kl}}\right]_1$	rad/m	partial derivatives of the angular rotation between wheel carrier *ij* and chassis with respect to the generalized coordinate q_{0kl}

The partial derivation $\vec{\Pi}_{2Rij,\, q_{0kl}}$ is not immediatly available. We obtain it from the partial derivation of the rotation vector $\frac{\partial \vec{r}_{ij}^{kin}}{\partial q_{0kl}}$ with the help of the Hamel transformation matrix H:

$$\left[\vec{\Pi}_{2Rij,\, q_{0kl}}\right]_1 = \left[H\right]\left[\frac{\partial \vec{r}_{ij}^{kin}}{\partial q_{0kl}}\right] = \begin{bmatrix} \cos(r_z) & -\sin(r_z)\cos(r_x) & 0 \\ \sin(r_z) & \cos(r_z)\cos(r_x) & 0 \\ 0 & \sin(r_x) & 1 \end{bmatrix}\left[\frac{\partial \vec{r}_{ij}^{kin}}{\partial q_{0kl}}\right] \qquad \textbf{(EQ 41)}$$

In the same manner the other generalized forces Q_{0FL}, Q_{0FR}, and Q_{0RR} are calculated.

The generalized left rack force $Q_{SteerFL}$ is obtained by calculating:

$$Q_{SteerFL} = \left[\vec{F}_{FL}^{WC}\right]_1 \cdot \left[\frac{\partial \vec{t}_{FL}^{kin}}{\partial q_{Steer}}\right]_1 + \left[\vec{T}(O_{2FL})^{WC}\right]_1 \cdot \left[\vec{\Pi}_{2RL,\, q_{Steer}}\right]_1 + \sum_X F_{FL}^{SElmX} \cdot \frac{\partial l_{FL}^{SElmX}}{\partial q_{Steer}} \qquad \textbf{(EQ 42)}$$

$$\text{with } \left[\vec{\Pi}_{2Rij,\, q_{Steer}}\right]_1 = \begin{bmatrix} \cos(r_z) & -\sin(r_z)\cos(r_x) & 0 \\ \sin(r_z) & \cos(r_z)\cos(r_x) & 0 \\ 0 & \sin(r_x) & 1 \end{bmatrix}\left[\frac{\partial \vec{r}_{ij}^{kin}}{\partial q_{Steer}}\right] \qquad \textbf{(EQ 43)}$$

In the same manner the generalized right rack force $Q_{SteerFR}$ is obtained.

Part 12: Update of trailer hitch forces/torques

In this function part the internal trailer model hitch forces and torques are copied in the vehicle environment. These variables, calculated in the function *Trailer_Calc* are:

State variable	Unit	Info
$\left[\vec{T}(O_H)^{hitch}\right]_0$	Nm	trailer hitch torques in the hitch point, expressed in global frame Fr_0
$\left[\vec{F}^{hitch}\right]_0$	N	trailer hitch forces in the hitch point, expressed in global frame Fr_0

Part 13: Update of loads masses/inertias

All additional vehicle loads are merged to one total load with one mass, position and inertia:

$$m_T^{load} = \sum_i m_i^{load} \quad \text{(EQ 44)}$$

$$\overrightarrow{O_1 M_T^{load}} = \frac{1}{m_T^{load}} \sum_i \overrightarrow{O_1 M_i^{load}} \cdot m_i^{load} \quad \text{(EQ 45)}$$

$$I_T^{load}(M_T^{load}) = \sum_i \left(I_i^{load}(M_i^{load}) + m_i^{load} \begin{bmatrix} (d_z^2 + d_y^2) & -d_x d_y & -d_x d_z \\ -d_x d_y & (d_x^2 + d_z^2) & -d_y d_z \\ -d_x d_z & -d_y d_z & (d_y^2 + d_x^2) \end{bmatrix} \right) \text{ with}$$

$$\begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} = \overrightarrow{M_T^{load} M_i^{load}} \quad \text{(EQ 46)}$$

Part 14: MESA VERDE dynamics calculation

In this function part the second set of MESA VERDE equations is called for the dynamics calculation. This part calculates the second order derivatives $\ddot{\boldsymbol{q}}_i$ of the generalized coordinates of the system:

$$\ddot{q}_{tx}, \ddot{q}_{ty}, \ddot{q}_{tz}, \ddot{q}_{rx}, \ddot{q}_{ry}, \ddot{q}_{rz}, \ddot{q}_{0FL}, \ddot{q}_{0FR}, \ddot{q}_{0RL}, \ddot{q}_{0RR}, \ddot{q}_{rxA2B}, \ddot{q}_{ryA2B}$$

depending on the generalized mass matrix $\boldsymbol{A}$ and the matrix $\boldsymbol{B}$ containing:

$$\vec{T}(O_1)^{transm}, \vec{F}_{ij}^{WC}, \vec{T}(O_{2ij})^{WC}, \vec{F}^{aero}, \vec{T}(O_A)^{aero}, \vec{T}(O_1)^{A2B}, Q_{0ij}, \vec{F}^{hitch}, \vec{T}(O_H)^{hitch}$$

Part 15: Integration of the generalized coordinates

In this calculation part all generalized coordinates $\boldsymbol{q}_i$ are determined with the aid of their first and second order derivatives using the forward Euler's method:

$$\dot{q}_i^{n+1} = \dot{q}_i^n + dt \cdot \ddot{q}_i^n \quad \text{(EQ 47)}$$

$$q_i^{n+1} = q_i^n + dt \cdot \dot{q}_i^n \quad \text{(EQ 48)}$$

Chapter 9

Vehicle Model 'Flexible'

9.1 Overview

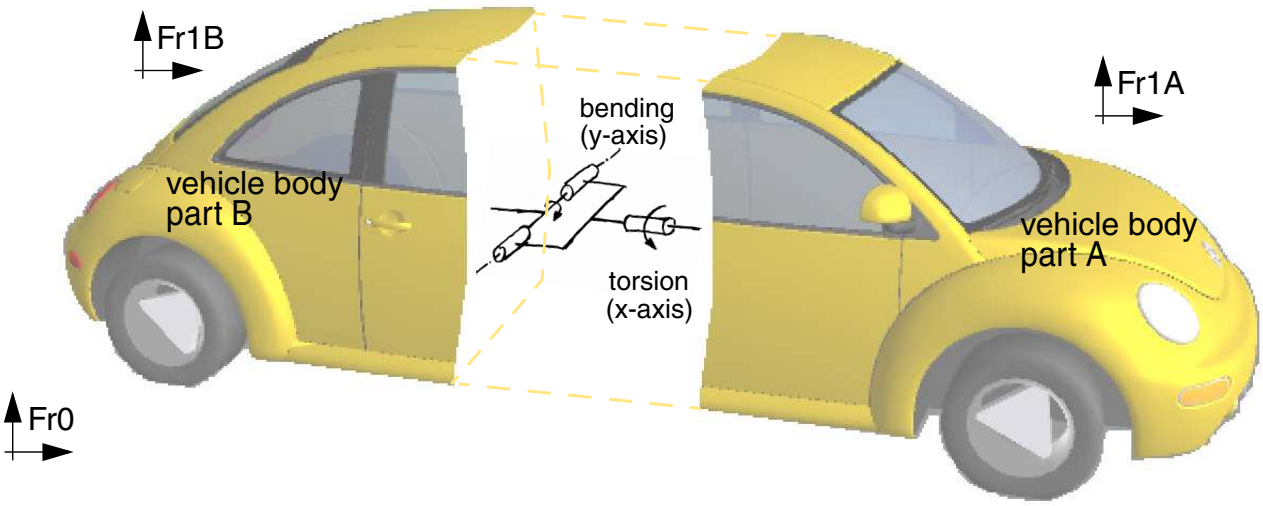


Figure 9.1: Structure of vehicle model Flexible

With the vehicle model "Flexible", torsion and bending of the vehicle body can be simulated. To realise this, the vehicle body frame of the generic vehicle model "Rigid" is divided in two frames Fr1A and Fr1B, which are connected by an X-Y rotational joint. This joint is modelled with a spring-damper element for the bending and the torsion axis. Fr1A corresponds to the frame Fr1 of the vehicle model "Rigid".

Using the vehicle model "Rigid" as well as "Flexible", the vehicle body is modelled with two body parts A and B. CarMaker gives the possibility to parametrise both parts A and B or only part A.

Using the vehicle model "Flexible" aerodynamic forces acts on part A and the trailer hitch is mounted on part B.

9.2 Parameters – Vehicle Parameter Set

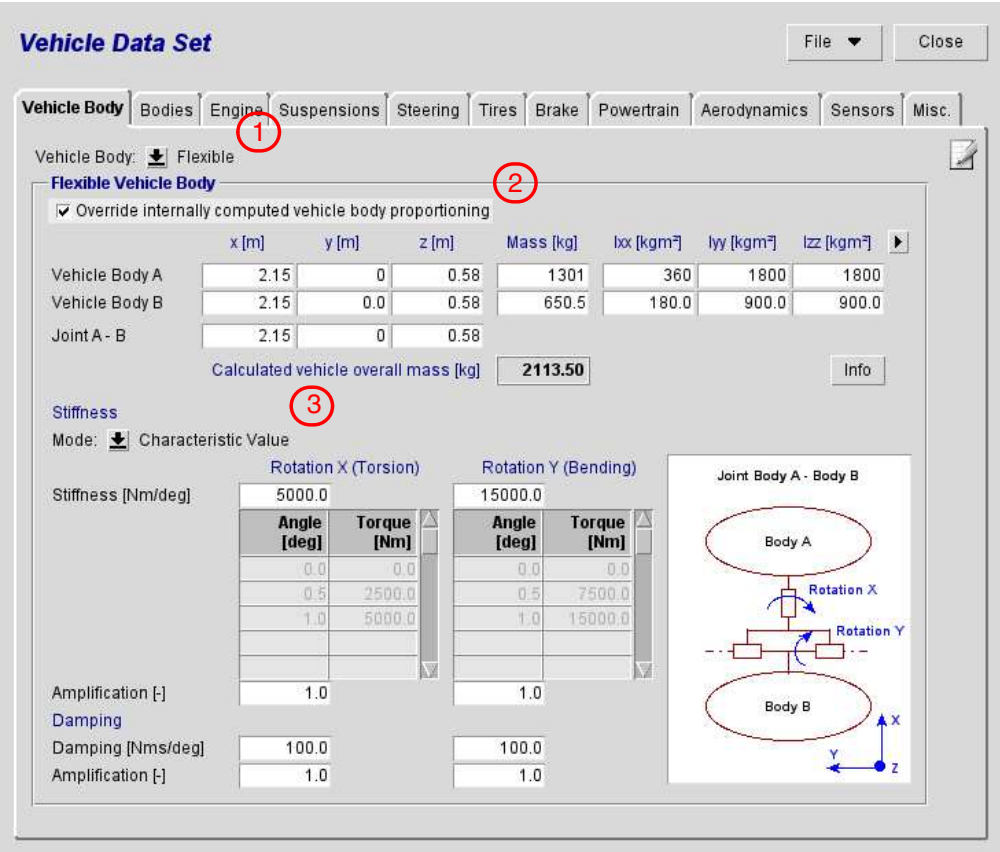


Figure 9.2: "Vehicle Data Set" in the GUI with Flexible

VehicleModel.Kind = *Kind*

This parameter selects the vehicle model (see also Figure 9.2 indication 1).
Kind = `FlexBody` for vehicle model with flexible body
Kind = `RigidBody` for the generic vehicle model without flexible body (default configuration)

VehicleModel.Mode= *Mode*

This mode is to select the vehicle body parametrisation (see Figure 9.2 indication 2).
Mode = `BodyA` :

- can be selected for the vehicle model "Flexible", default for model "Rigid"
- body mass and inertia are parametrised and these mass and inertia are separated similarly: a half for the part A and a half for the part B (the new position of the masses is chosen so that the total inertia, mass and CoG stay unchanged)

Mode = `BodyA+B`:

- can be selected only for the vehicle model "Flexible"
- mass, inertia and the position of the body A and B are parametrised independently

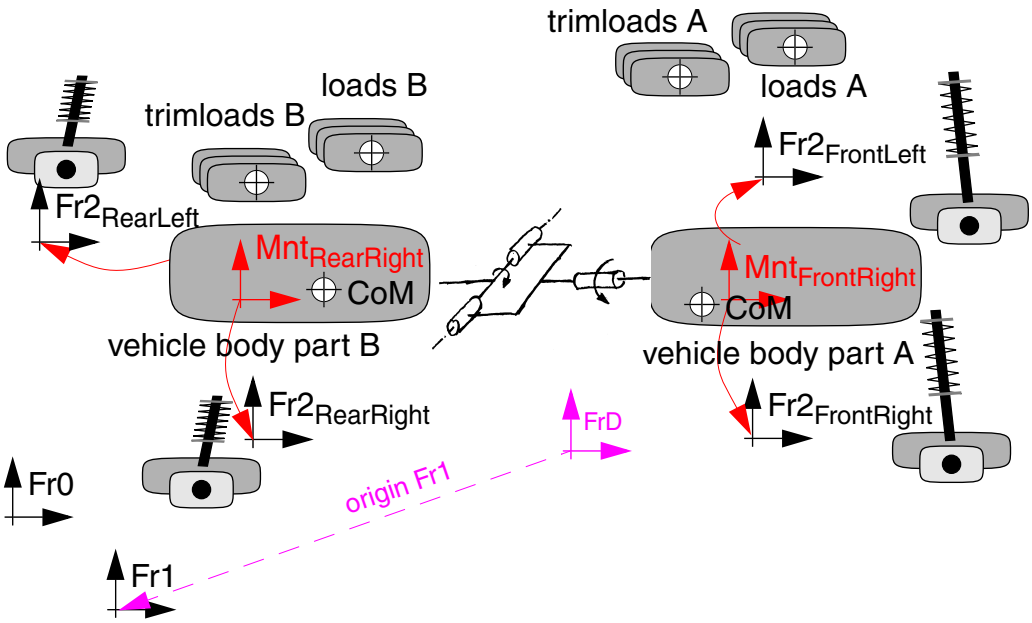


Figure 9.3: Detailed structure of vehicle model Flexible

Description of the bodies and other things:

General	Part A	Part B	Parameter
vehicle body			Body
	x		BodyA
		x	BodyB
trimloads	x	x	TrimLoad.*
vehicle loads	x	x	VehicleLoad.*

Bodies and Masses (Parameters in the Vehicle Data File)

Body.pos
Body.mass
Body.I

Vehicle body: Its center of mass is placed in the design frame at *Body.pos*. The vehicle body has the mass *Body.mass* and the inertia tensor *Body.I*.

A vehicle body with inertia I_{xx}=360 kg*m^2, I_{yy}=1800 kg*m^2, I_{zz}=1800 kg*m^2:

```
Body.pos =   2.15 0.0 0.58
Body.mass =  1500.0
Body.I =     360 1800 1800
```

This parameter is only used for the *Mode* = `BodyA`. If this parameter is used, it is divided into two bodies: a half for part A and a half for part B.

BodyA.pos
BodyA.mass
BodyA.I

Its center of mass is placed in the design frame at *BodyA.pos*. The vehicle body has the mass *BodyA.mass* and the inertia tensor *BodyA.I*.

A vehicle body A with inertia I_{xx}=180 kg*m^2, I_{yy}=500 kg*m^2, I_{zz}=500 kg*m^2:

```
BodyA.pos =  3.15 0.0 0.58
BodyA.mass = 750.
BodyA.I =    180. 500 500
```

This parameter is only used for the *Mode* = `BodyA+B`. Don't use this parameter together with parameter *Body*.

BodyB.pos
BodyB.mass
BodyB.I

Its center of mass is placed in the design frame at *BodyB.pos*. The vehicle body has the mass *BodyB.mass* and the inertia tensor *BodyB.I*.

A vehicle body Bwith inertia I_{xx}=180 kg*m^2, I_{yy}=1000 kg*m^2, I_{zz}=1000 kg*m^2:

```
BodyB.pos =     1.15 0.0 0.58
BodyB.mass =    750.
BodyB.I =       180. 1000 1000
```

This parameter is only used for the *Mode* = `BodyA+B`. Don't use this parameter together with parameter *Body*.

Remark

Don't use the parameter *Body* together with parameter *BodyA/BodyB*. This body is read in, and initializes *BodyA* and *BodyB*. If *BodyA* and/or *BodyB* are given too, they overwrite the first values.

Flexibility – Joint PartA/PartB

Flex.JointFr1Fr1B.Kind = ***Kind***

The body flexibility can be configured in different types (see Figure 9.2 indication 3):
Kind = `Coeff`: by spring and damper coefficients
Kind = `TrqLin`: by spring characteristic (rotation – torque) and damper coefficients

Flex.JointFr1Fr1B.pos = ***x y z***

Position of joint to vehicle body part B (frame 1 B, Fr1B).

Flex.JointFr1Fr1B.k.x = ***dTrqx_drx***
Flex.JointFr1Fr1B.k.x:

0.0	***0.0***
rx	***Trq_x***
rx	***Trq_x***
...	

Vehicle body spring characteristics for rotation along body X axis. Unit: Nm/deg.
Variant a) `Coeff`: stiffness coefficient.
Variant b) `TrqLin`: spring mapping, torque as function of rotation angle. The mapping is extrapolated with last interval gradient.

Flex.JointFr1Fr1B.k.y = ***dTrqy_dry***
Flex.JointFr1Fr1B.k.y:

0.0	***0.0***
ry	***Trq_y***
ry	***Trq_y***
...	

Vehicle body spring characteristics for rotation along body Y axis. Unit: Nm/deg.
Variant a) `Coeff`: stiffness coefficient
Variant b) `TrqLin`: spring mapping, torque as function of rotation angle. The mapping is extrapolated with last interval gradient.

Flex.JointFr1Fr1B.k.x.Amplify = ***AmpliFactor.x***
Flex.JointFr1Fr1B.k.y.Amplify = ***AmpliFactor.y***

Optional. Stiffness amplification factor. Default is 1.0.

Flex.JointFr1Fr1B.d.x = ***dTrqx_drvx***
Flex.JointFr1Fr1B.d.y = ***dTrqy_drvy***

Vehicle body damping. Damping coefficients for X and Y rotation.
This parameters has to be greater than zero, to damp torsional motion.
Unit: Nms/deg.

Flex.JointFr1Fr1B.d.x.Amplify = ***AmpliFactor.x***
Flex.JointFr1Fr1B.d.y.Amplify = ***AmpliFactor.y***

Optional. Damping amplification factor. Default is 1.0.

9.3 Parameters – Test Run Parameter Set

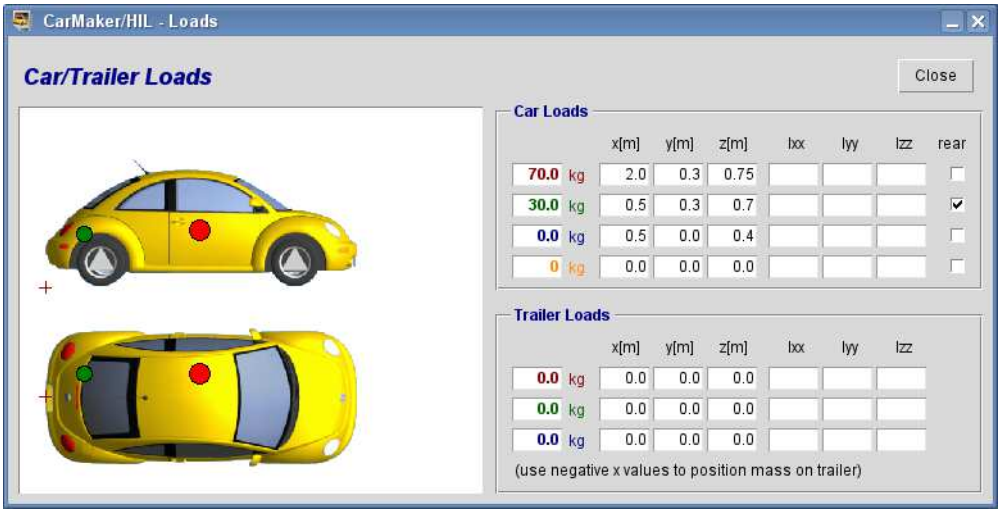


Figure 9.4: "Car/Trailer

VehicleLoad.*No*.mass =	***Mass***		**[*IsActive*]**			
VehicleLoad.*No*.I =	***Ixx***	***Iyy***	***Izz***	**[*Iyz***	***Ixz***	***Ixy*]**
VehicleLoad.*No*.pos =	***x***	***y***	**z**			
VehicleLoad.*No*.mounted =	**[*MountFrame*]**					

Four movable vehicle body mounted loads, *No* = 0,1,2,3, defined in the test run parameter set (see Figure 9.4). The vehicle load has the mass *Mass* and the inertia tensor *Ixx, Iyy, Izz* etc, see *section 7.3.1 'Overview Mass Geometry' on page 84*. It can be deactivated by *IsActive*=0, optional, default *IsActive*=1. The center of mass is at *x, y, z* in the frame Fr1.

The vehicle load can be mounted on *MountFrame* = Fr1A or Fr1B, default is vehicle body frame Fr1A. Using the GUI, the vehicle load is mounted on *MountFrame* = Fr1B if the button "rear" in Figure 9.4 is activated.

9.4 Examples

Example 1:

```
VehicleModel.Kind =          RigidBody
VehicleModel.Mode =          BodyA

Body.pos =          2.15   0.0    0.58
Body.mass =         1500
Body.I =            360 1800 1800
```

Example 2:

```
VehicleModel.Kind =          FlexBody
VehicleModel.Mode =          BodyA+B

BodyA.pos =         3.15   0.0    0.580
BodyA.mass =        750
BodyA.I =           180 900 900

BodyB.pos =         1.15   0.0    0.580
BodyB.mass =        750
BodyB.I =           180 900 900

Flex.JointFr1Fr1B.pos =          2.15   0.0    0.580
Flex.JointFr1Fr1B.Kind =         Coeff

Flex.JointFr1Fr1B.k.x.Amplify =  1.0
Flex.JointFr1Fr1B.k.x =          15000
Flex.JointFr1Fr1B.k.y.Amplify =  1.0
Flex.JointFr1Fr1B.k.y =          15000

Flex.JointFr1Fr1B.d.x.Amplify =  1.0
Flex.JointFr1Fr1B.d.x =          100
Flex.JointFr1Fr1B.d.y.Amplify =  1.0
Flex.JointFr1Fr1B.d.y =          100
```

9.5 User Accessible Quantities

Name	Frame	Unit	Info
Car.Fr1B.Trq_B2A.x Car.Fr1B.Trq_B2A.y	Fr1A	Nm	spring-damper torque from Fr1B to Fr1A
Car.Fr1B.Trq_B2A.x_ext Car.Fr1B.Trq_B2A.y_ext	Fr1A	Nm	external torque from Fr1B to Fr1A
Car.Fr1B.rax Car.Fr1B.ray	–	rad/s^2	rotational accelerations of body flexibility joint
Car.Fr1B.rvx Car.Fr1B.rvy	–	rad/s	rotational velocity of body flexibility joint
Car.Fr1B.rx Car.Fr1B.ry	–	rad	rotation angles of body flexibility joint
Car.FrX.Grad	–		
Car.FrX.Slope	–		
Car.ConA.Pitch_X	FrX		pitch angle of Fr1A
Car.ConA.Roll_X	FrX		roll angle of Fr1A
Car.ConB.Pitch_X	FrX		pitch angle of Fr1B
Car.ConB.Roll_X	FrX		roll angle of Fr1B
FrX is a frame, defined by the tire road contact points			

Chapter 10

Elastically mounted bodies

10.1 Engine

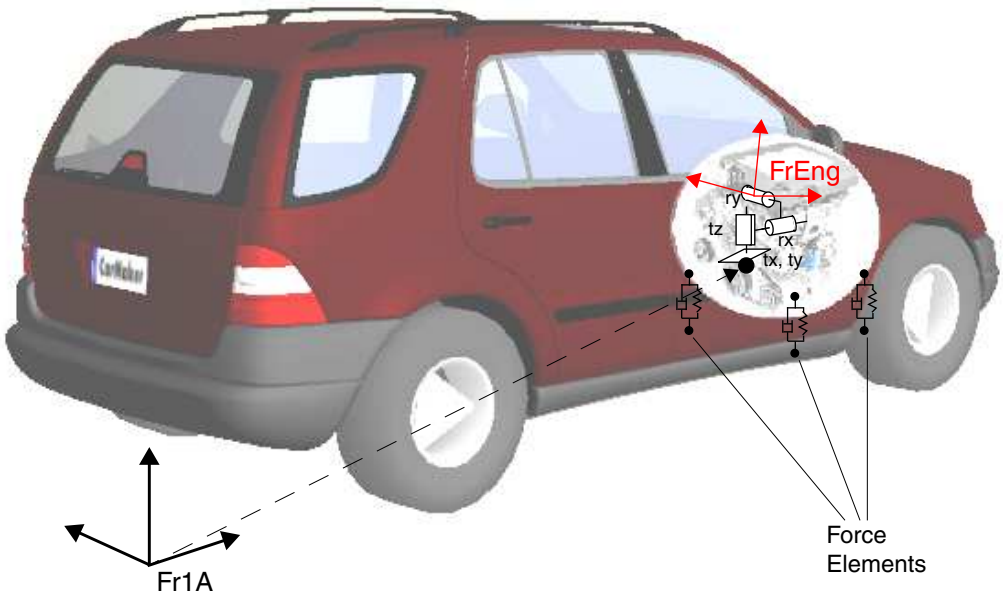


Figure 10.1: Vehicle engine modelling

The engine is an elastically supported body on the vehicle main body. This body is fixed to the engine system called *FrEng*. The motion of the engine frame *FrEng* relative to the vehicle frame *Fr1A* is defined by a joint with five degrees of freedom: longitudinal, lateral and vertical displacement tx,ty,tz; rotation along x-axis rx and rotation along y-axis ry.
There are two different ways to define the engine mount force elements: force/torque elements located in the joint (generalized forces) or three until four free located mount points with force element like shown in Figure 10.1.

10.1.1 Parameters

General Parameters

Eng.Active = *bool*

Indicates if the elastically mounted engine is activated. Default: 0.

Eng.Joint.pos = *x y z*

Specifies the engine joint position (origin of `FrEng`) given in `FrD` (design configuration axis system).

Eng.Bdy.pos = *x y z*
Eng.Bdy.mass = *Mass*
Eng.Bdy.I = ...

Engine body with total mass *Eng.Bdy.mass* [kg] and the inertia tensor *Eng.Bdy.I* [kgm^2]. Position of the center of mass (CoM) *Eng.Bdy.pos* is given in `FrD` (design configuration axis system).

Force Element Parameters

Eng.Kind = *KindStr*

Specifies the kind of the mounting suspension. Possible values are:

- *GenFrc*: Force elements in the joint (generalized forces): forces FrcX, FrcY, FrcZ; torques TrqX and TrqY. This is the default.
- *3xFrc*: three free located mount points with force element.
- *4xFrc*: four free located mount points with force element.

For the following parameters the prefix *SuspPre* is taken depending on the suspension kind:

- *GenFrc*: *SuspPre*=["Frc_X"; "Frc_Y"; "Frc_Z"; "Trq_X"; "Trq_Y"]
- *3xFrc ... 4xFrc*: *SuspPre*=["Frc.0.X"; "Frc.0.Y"; "Frc.0.Z"; ... "Frc.3.Z"]

Eng.<SuspPre>.Kind = *KindStr*

Specifies the kind of mount suspension force element. Possible values are:

- *Coeff*: Spring stiffness and damping are defined with a constant characterictic value. This is the default.
- *Curve*: Spring and damping forces are described with a non-linear 1D look-up table.

- *DVA*: The desired force element force is set using the Direct Variable Access on the quantities:
  - *GenFrc*: *Car.Eng.GenFrc.[x,y,z], Car.Eng.GenTrq.[x,y]*
  - *3xFrc ... 4xFrc*: *Car.Eng.Frc.[0..3].Frc.[x,y,z]*
    For the static equilibrium the forces are calculated with the model kind *Coeff*.
- *User*: The desired force element force is calculated by an user implemented function, as shown in the following example:

Example

```
In User.c:
static double MyCalcGenFrcZ (double dt, double q, double qp) {
double GenFrcZ;
GenFrcZ = ...;   //place here your model
return GenFrcZ;
}
int User_Register (void) {
int FrcElId = 2;  //2=FrcZ; (description of FrcElId in include/Car.h)
Set_UserEngFrcElFunc (MyCalcGenFrcZ, FrcElId);
}
```

Eng.<SuspPre>.Spring.c = Stiffness
Eng.<SuspPre>.Spring.Data : *Look-Up Table*

The spring characteristic can be defined with a constant stiffness ([N/m or Nm/rad]) or with a look-up table describing the non-linear spring force/torque (N=f(m) or Nm=f(rad)).

Eng.<SuspPre>.Spring.Amplify = *Value*

Spring amplification factor. Default: 1.0.

Eng.<SuspPre>.Spring.x0 = *UnstretchedLength / UnstretchedAngle*

Unstretched spring length (angle for rotational spring). Unit: [m] or [rad].
Default: for the kind *GenFrc* the length is determined to get the generalized coordinate nearly zero; for the kind *3xFrc ... 4xFrc* the length is 0.1 for the Z component, otherwise zero.

Eng.<SuspPre>.Damper.d = Damping
Eng.<SuspPre>.Damper.Data : *Look-Up Table*

The damper characteristic can be defined with a constant ([Ns/m or Nms/rad]) or with a look-up table describing the non-linear damper force/torque (N=f(m/s) or Nm=f(rad/s)).

Eng.<SuspPre>.Damper.Amplify = *Value*

Damper amplification factor. Default: 1.0.

Eng.<Frc.[0...3]>.pos = *x* *y* *z*

Specifies the position of the suspension lower mounting point given in `FrD` (design configuration axis system). This parameter is only used for the kind *3xFrc ... 4xFrc.*

10.1.2 User Accessible Quantities

Please refer to section 24.7.4 'Elastic mounted Engine'.

Chapter 11

Suspension Force Elements

The contribution of each suspension force element results in the wheel contact force. Four types of force elements are modeled:

- the suspension spring (Spring, SecSpring),
- the suspension damper (DampPull, DampPush, DamperTopMount),
- the suspension buffer (BufPull, BufPush) and
- the stabilizer or anti-roll bar or stabilizer bar (Stabi).

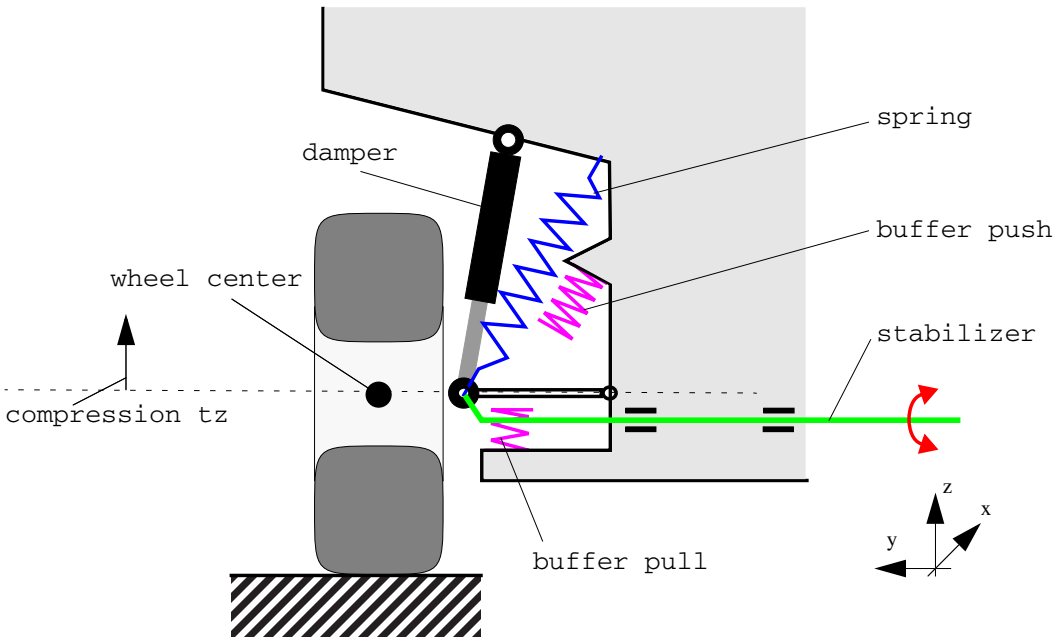


Figure 11.1: Force elements shown at the left wheel

11.1 External Suspension Forces

External suspension forces are added to the forces generated by the CarMaker built in models for spring/bumpers, damper, stabilizer and buffer:

$$
\begin{aligned}
F_{SpringTot} &= F_{Spring} + F_{SpringExt} \\
F_{DampTot} &= F_{Damp} + F_{DampExt} \\
F_{StabiTot} &= F_{Stabi} + F_{StabiExt} \\
F_{BufferTot} &= F_{Buffer} + F_{BufferExt}
\end{aligned}
\qquad \text{(EQ 49)}
$$

There is multiple usage for external suspension forces (according to (EQ 49)):

- Replace suspension forces by external forces. This is used when a custom suspension model (e. g. Simulink model) should replace the CarMaker built in model for some or all suspension forces. The original forces have to be set to zero using the “amplify” parameters (*<Pre>.XXX.Amplify* = 0).
- Add forces to the forces calculated by the built in model. This can be used to simulate forces from an active suspension device which are superimposed to the conventional forces.

SuspExtFrcs.FName = ***FileName*** **or**
SuspExtFrcs.Kind = ***ModelKind***

The model is selected by the Kind entry *ModelKind.* Instead of using the Kind key, an external parameter set can be referenced by its filename *FileName*. *FileName* is the path to the miscellaneous directory (project directory > Data > Misc).

General Remarks

- Suspension models have to be registered by the CarMaker model management mechanism. Each model needs
  - an unique kind key to reference exactly to this very model from the pool of suspension models.
  - interface functions to initialize, calculate and delete a model instance.
- All parameters other than (**.FName* and **.Kind*) are model specific. Their keys have to start with the prefix "*ExtSuspFrcs.*"
- The amplification parameters **.Spring_ext.Amplify*, **.Damp_ext.Amplify* and **.Stabi_ext.Amplify*, used in CarMaker versions before 2.1, are not available any longer. In case their functionality is required, it has to implemented in the external suspension model code itself.

11.1.1 General Parameters

SuspExtFrcs.Kind = ***KindStr*** ***VersionId***

With this parameter a model for the Suspension Forces is selected. The Suspension Forces library provides the following models:

KindStr	Description
SuspComponent	Model with four submodels (spring, damper, stabilizer, buffer).

Example `SuspExtFrcs.Kind = SuspComponent 1`

11.1.2 Model "SuspComponent"

Every suspension force element (spring, damper, stabilizer and buffer) can be chosen independent from the others. The user can create custom suspension models (e. g. Simulink model) to specify the force elements.

SuspExtFrcs.Spring.Kind = ***KindStr*** ***VersionId***
SuspExtFrcs.Damper.Kind = ***KindStr*** ***VersionId***
SuspExtFrcs.Stabi.Kind = ***KindStr*** ***VersionId***
SuspExtFrcs.Buffer.Kind = ***KindStr*** ***VersionId***
or
SuspExtFrcs.Spring.FName = ***FileName***
SuspExtFrcs.Damper.FName = ***FileName***
SuspExtFrcs.Stabi.FName = ***FileName***
SuspExtFrcs.Buffer.FName = ***FileName***

The model is selected by the Kind entry *ModelKind.* Instead of using the Kind key, an external parameter set can be referenced by its filename *FileName*. *FileName* is the path to the miscellaneous directory (project directory > Data > Misc). The first line in the external parameter file has to have the "FileIdent" entry (see section 2.2.1 'File identification'):

Infofile	FileIdent
Spring	CarMaker-SuspEF_Spring-<Kind> <VersionID>
Damper	CarMaker-SuspEF_Damper-<Kind> <VersionID>
Stabi	CarMaker-SuspEF_Stabi-<Kind> <VersionID>
Buffer	CarMaker-SuspEF_Buffer-<Kind> <VersionID>

Example `FileIdent =CarMaker-SuspEF_Buffer-MyBuffer 1`

If the KindStr is blank, this suspension force element will not be computed, but the others will still be used.

Example
```
SuspExtFrcs.Spring.Kind =
SuspExtFrcs.Damper.Kind = <Kind> <VersionID>
SuspExtFrcs.Stabi.FName = <FileName>
SuspExtFrcs.Buffer.Kind =
```

11.2 Springs

11.2.1 Coil Spring

This module simulates a conventional spring suspension. It calculates the spring component force F_{Spring}.

As shown in Figure 11.3 there is a relation between the deflection of the wheel and the spring. There is a translation between the forces F_z and F_{Spring} and between the travel of the wheel t_z and the spring x^*.

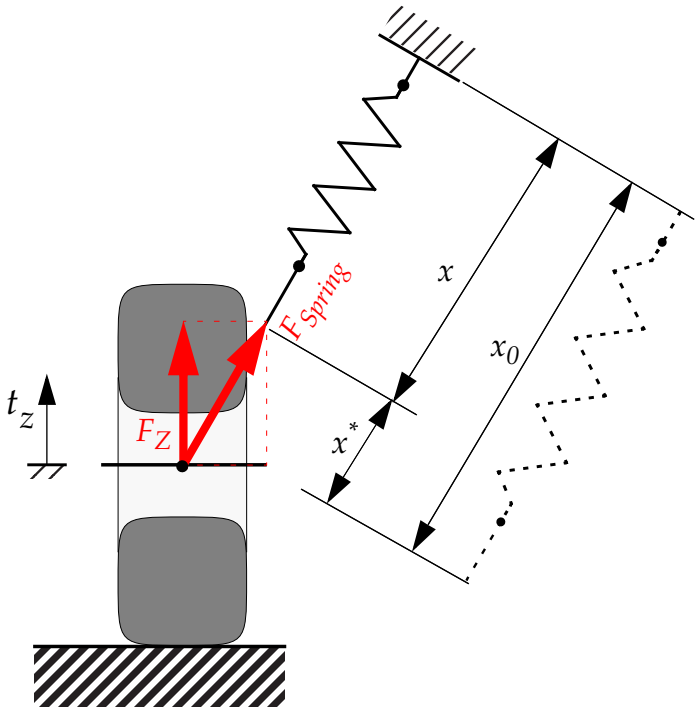


Figure 11.3: Calculation of spring forces

When the suspension gets compressed the spring length x is decreasing and the resulting spring force is increasing. To accommodate this the following calculation is applied:

$$\begin{aligned} F_{\text{Spring}} &= \text{amp} \cdot f(x^*) \\ x^* &= -(x - x_0) \end{aligned} \quad . \qquad \textbf{(EQ 50)}$$

The factor `amp` in (EQ 50) can be used to modify the spring forces by a given factor. Usually this is for test purposes only and the factor should remain set to one.

The quantity x_0 is called relaxed (or unstretched) length of the spring. The resulting spring force depends on the difference between the relaxed length x_0 and the current length x. The current length is the distance between the lower and upper attachment point of the spring.

According to (EQ 50) the calculation of x^* needs the current spring length x. It is obtained from the suspension kinematics module according to Figure 11.4. q represents the generalized coordinates of the suspension.

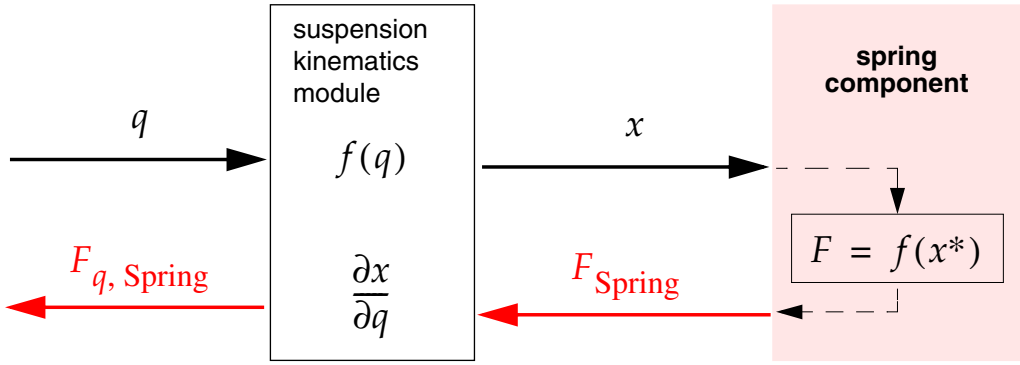


Figure 11.4: Transformation of spring deflections and forces

If *no* spring length is parametrized, minus wheel compression tz is used.

Parameters

SuspF means front axle, SuspR means rear axle.

.Spring.Amplify = *Factor

Spring amplification factor *Factor*.

Spring.l0 = *UnstressedLength

Optional. Unstressed (i.e. force free) length *UnstressedLength* of the spring, the unit is m.

.Spring = *ConstStiffness
.Spring: *DataTable

The spring characteristic can be defined with a constant or with a data table describing the non-linear spring behavior.
This characteristic translates compression (x^* from (EQ 50)) to spring force.

11.2.2 Secondary Spring

This module calculates a secondary spring force $F_{SecSpring}$ generated by the suspension bushing torsion torques along the bushing rotation axle while wheel travel.

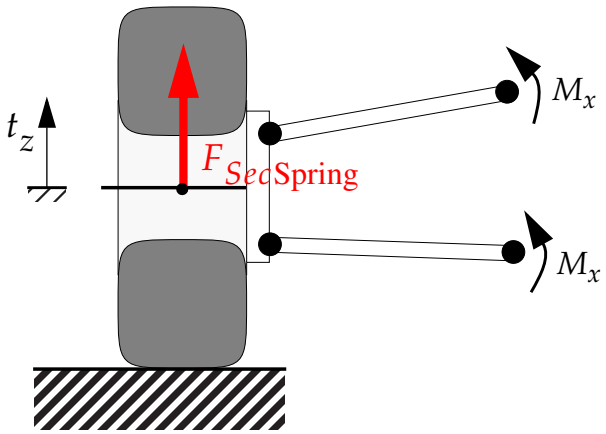


Figure 11.5: Calculation of secondary spring forces

The relation between the secondary force $F_{SecSpring}$ and the wheel travel t_z can be described as a linear or a non-linear function:

$$F_{SecSpring} = \text{amp} \cdot f(tz) \qquad \textbf{(EQ 51)}$$

The factor `amp` can be used to modify the spring forces by a given factor. Usually this is for test purposes only and the factor should remain set to one.

Parameters

SuspF means front axle, SuspR means rear axle.

.SecSpring.Amplify = *Factor

Secondary spring amplification factor *Factor*.

.SecSpring = *ConstStiffness
.SecSpring: *DataTable

The secondary spring characteristic can be defined with a constant or with a data table describing the non-linear spring behavior. This characteristic translates wheel travel t_z to secondary spring force $F_{SecSpring}$.

11.3 Dampers

11.3.1 Damper

This module simulates a suspension damper. It calculates the force F_{Damp}.

The suspension dampers force F_{Damp} depends on the velocity $\dot{x}$. Different characteristics are taken into account by defining different functions for “pull” ($\dot{x} > 0$) and “push” ($\dot{x} \leq 0$).

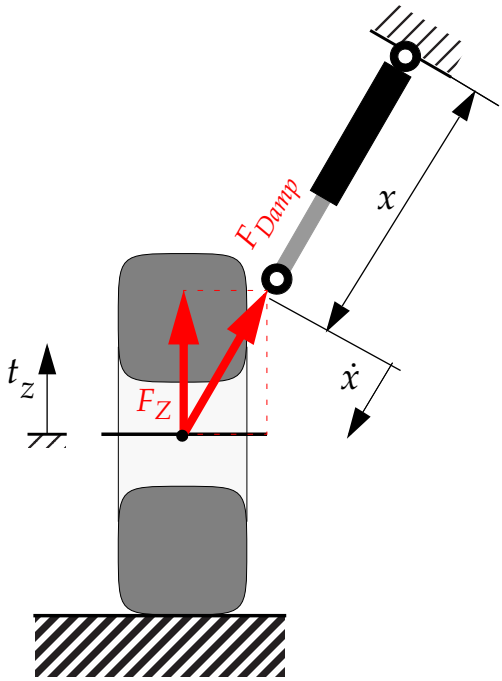


Figure 11.6: Calculation of damper forces

An increasing length x of the damper leads to a positive velocity $\dot{x}$.

$$\begin{aligned} F_{\mathrm{Damp}}(\dot{x}) &= \mathrm{amp} \cdot F_{\mathrm{DampPull}}(\dot{x}) \qquad \dot{x} > 0 \\ F_{\mathrm{Damp}}(\dot{x}) &= \mathrm{amp} \cdot F_{\mathrm{DampPush}}(\dot{x}) \qquad \dot{x} \leq 0 \end{aligned} \quad . \qquad \textbf{(EQ 52)}$$

The `amp` factor in (EQ 52) can be used to modify the results of the calculation by a given factor. Usually this is for test purposes only and the factor should remain set to one.

The dampers pull/push characteristics are defined separately. Pull is when the damper is getting longer, push is when the damper is getting shorter.

The velocity $\dot{x}$ is calculated by differentiation of the damper length x. It is obtained from the suspension kinematics module according to Figure 11.7. q represents the generalized coordinates of the suspension.

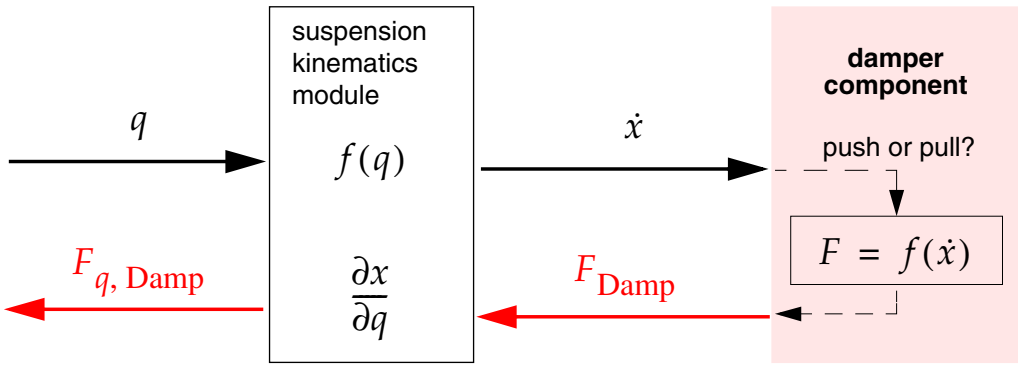


Figure 11.7: Transformation of damper deflections and forces

If *no* damper length is parametrized, minus wheel compression tz is used.

All characteristics have to be defined in first quadrant. CarMaker takes care of the correct signs.

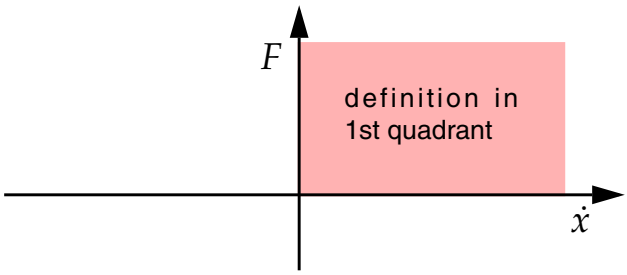


Figure 11.8: Definition of damper push and pull characteristics

Parameters

.Damp_Push.Amplify = *Factor

Damper push force amplification factor *Factor* (see (EQ 52)).

.Damp_Push = *ConstStiffness
.Damp_Push: *DataTable

The damper characteristic can be defined with a constant or with a data table describing the non-linear damper behavior.
This characteristic translates compression velocitiy $\dot{x}$ to damper push force. It has to be defined in first quadrant (positive velocities, positive forces). $F_{\mathrm{Damp}}(0) = 0$ should be obeyed. The table has two columns, velocity in m/s in the first, the damper forces in N the second.

Example

```
SuspF.Damp_Push:
        0       0
        0.05    90
        0.13    140
        0.26    250
        0.39    330
        0.52    410
        1.04    790
```

Example

```
SuspF.Damp_Push = 800
```

.Damp_Pull.Amplify = *Factor

Damper push force amplification factor *Factor* (see (EQ 52)).

.Damp_Pull = *ConstStiffness
.Damp_Pull: *DataTable

The damper characteristic can be defined with a constant or with a data table describing the non-linear damper behavior.
This characteristic translates extention velocitiy $\dot{x}$ to damper pull force. It has to be defined in first quadrant (positive velocities, positive forces). $F_{\mathrm{Damp}}(0) = 0$ should be obeyed. The table has two columns, velocity in m/s in the first, the damper forces in N the second.

Example

```
SuspF.Damp_Pull:
        0       0
        0.05    110
        0.13    350
        0.26    650
        0.39    725
        0.52    800
        1.04    1150
```

Example

```
SuspF.Damp_Pull = 800
```

11.3.2 Damper with Top Mount

Optionally a top mount can be simulated in addition to the suspension damper. It is then taken into account when the damper force F_{Damp} is calculated.

The top mount acts in direction of the usual damping force. It is modeled as Kelvin-Voigt element composed of a spring and a damper.

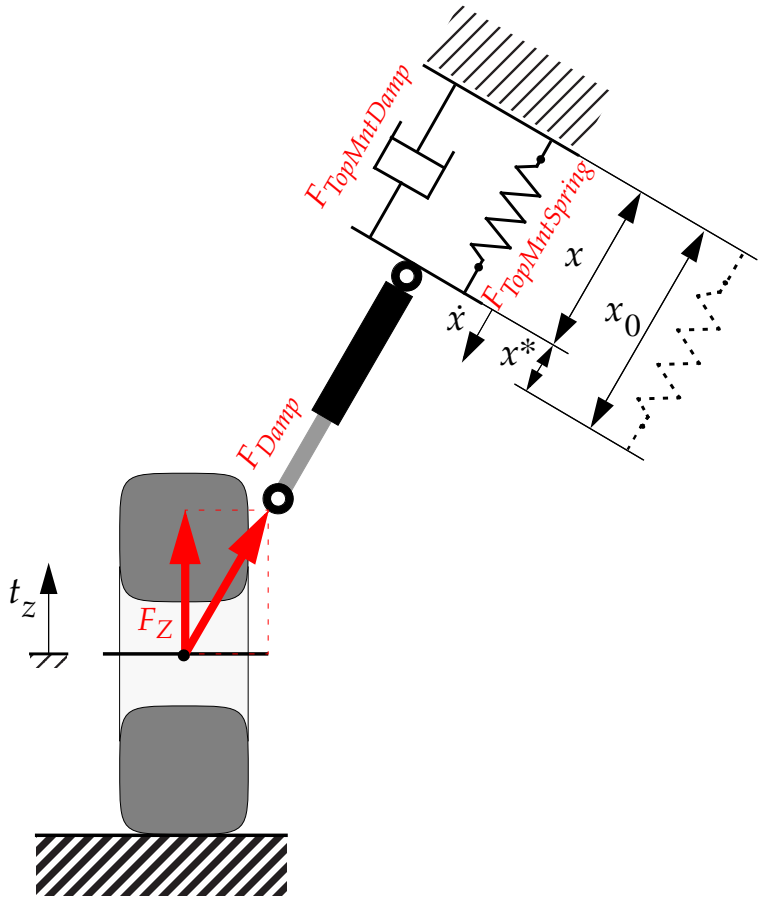


Figure 11.9: Calculation of damper forces with top mount

The quantity x is the distance between the lower and upper attachment point of spring and damper and therefore denotes their current length, whereas x_0 is the relaxed (or unstretched) length of the spring. $\dot{x}$ terms the compression/extension velocity of top mount.

For the simulation of top mount a linear model is used. The top mount force, which is equal to the damper force F_{Damp}, is calculated as

$$F_{\mathrm{TopMnt}}(\dot{x}, x^*) = F_{\mathrm{TopMntDamp}}(\dot{x}) + F_{\mathrm{TopMntSpring}}(x^*) \qquad \textbf{(EQ 53)}$$

$$\text{with} \quad F_{\mathrm{TopMntDamp}}(\dot{x}) = -d \cdot \dot{x} \qquad \textbf{(EQ 54)}$$

$$\text{and} \quad \begin{aligned} F_{\mathrm{TopMntSpring}}(x^*) &= k \cdot x^* \\ x^* &= -(x - x_0) \end{aligned} . \qquad \textbf{(EQ 55)}$$

All characteristics of top mount have to be defined in first quadrant. CarMaker takes care of the correct signs as it does for the damper push and pull characteristics.

Parameters

*.Damper.TopMnt.On = *bool*

Parameter for the activation of top mount. Default: 0.

*.Damper.TopMnt.d = *ConstDamping*

The damper characteristic of top mount is defined by a constant. Unit: [Ns/m].

Example `SuspF.Damper.TopMnt.d = 1000.0`

*.Damper.TopMnt.k = *ConstStiffness*

The spring characteristic of top mount is defined by a constant. Unit: [N/m].

Example `SuspF.Damper.TopMnt.k = 1000000.0`

11.4 Buffers / Bumpers

This module simulates suspension push/pull buffers. F_{BufPull} is the force of the *lower bump-stop*, F_{BufPush} is the force of the *upper bump-stop*.

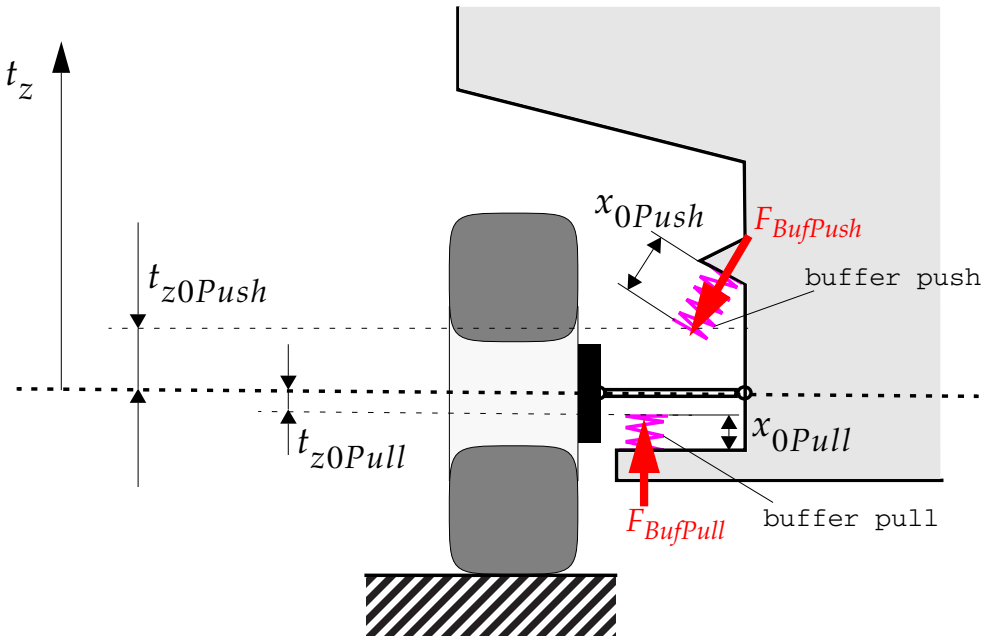


Figure 11.10: Buffer elements shown at left wheel

The suspension *buffer* forces F_{BufPull}, F_{BufPush} depend on the buffer compression x:

$$F_{\mathrm{BufPush}} = \mathrm{amp} \cdot f(x^*) \qquad x^* = x_{0,\,\mathrm{Push}} - x \qquad x^* > 0$$
$$F_{\mathrm{BufPush}} = 0 \qquad x^* < 0 \tag{EQ 56}$$

$$F_{\mathrm{BufPull}} = \mathrm{amp} \cdot (-f)(x^*) \qquad x^* = x - x_{0,\,\mathrm{Pull}} \qquad x^* > 0$$
$$F_{\mathrm{BufPull}} = 0 \qquad x^* < 0 \tag{EQ 57}$$

$x_{0,\,\mathrm{Push}}$, $x_{0,\,\mathrm{Pull}}$ are the relaxed (or unstretched) length of the upper/lower bump-stop.

For the calculation of the buffer forces according to (EQ 56) and (EQ 57) the current buffer compression x is needed. It is obtained from the suspension kinematics module according to Figure 11.11. q represents the generalized coordinates of the suspension.

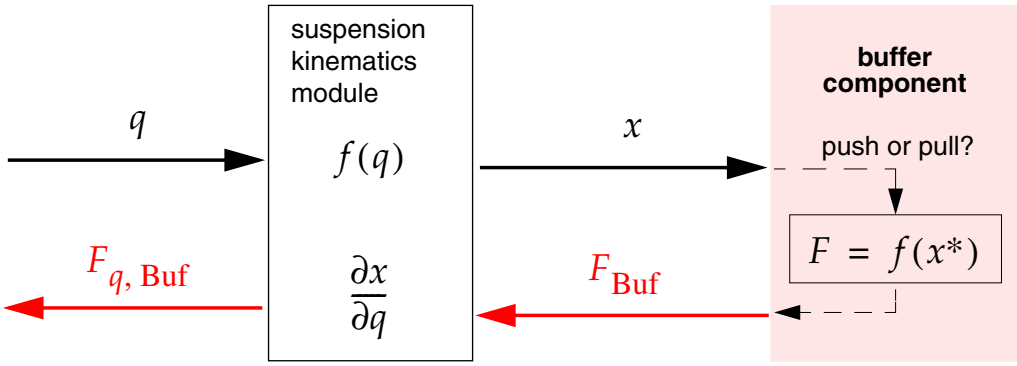


Figure 11.11: Transformation of buffers deflections and forces

If *no* buffer length is parametrized, minus wheel compression tz is used.

The following figure illustrates the calculation of the buffer compression from the non-linear kinematics file:

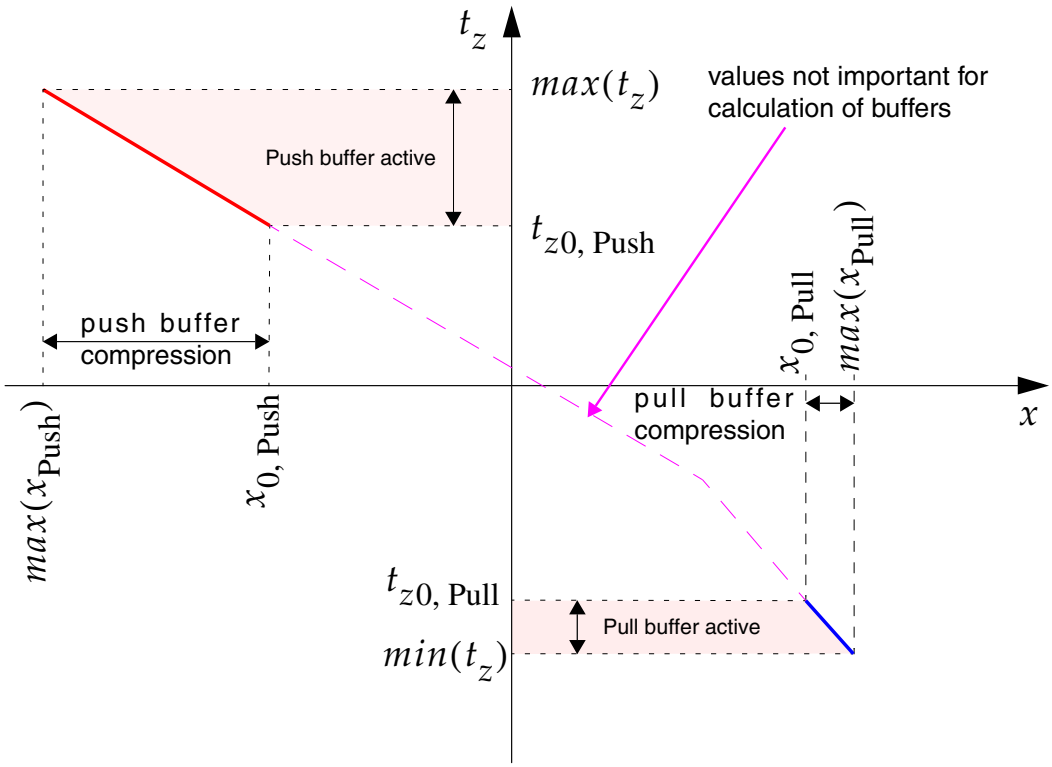


Figure 11.12: Calculating the buffer length out of kinematic data

The two parameters $t_{z0,\ Push}$ and $t_{z0,\ Pull}$ have the corresponding buffer compression zero offset values $x_{0,\ Push}$ and $x_{0,\ Pull}$ .The absolute values are unimportant since there is always the difference calculated from the current compression value to its zero offset.

Of importance is that the gradient of the compression curve is always negative, so that according to (EQ 56) and (EQ 57) a positive buffer compression is calculated. Values given between $x_{0,\ Push}$ and $x_{0,\ Pull}$ are ignored from CarMaker and therefore are not important.

The `amp` factor in (EQ 56) and (EQ 57) can be used to to modify the results of the calculation function by a given factor. Usually this is for test purposes only and the factor should remain set to one.

The buffers pull/push characteristics are defined separately.

If two buffers in parallel or series are used to parametrize a push or pull buffer, a replacement buffer is calculated including all parameters.

Parameters

SuspF means front axle, SuspR means rear axle

.Buf_Push.Amplify = *value
.Buf_Push.1.Amplify = *value

Push Buffer amplification factor.

Example
```
SuspF.Buf_Push.Amplify = 1.0
```

***.Buf_Push.tz0 = *value* [parallel]**
***.Buf_Push.1.tz0 = *value* [parallel]**

Push buffer position (defines ride clearance). According to (EQ 56) and Figure 11.10 the push buffer only acts for vertical displacements tz (of wheel center) greater than tz0(=x_{0Push}).
Unit is meters.
The option “parallel” means “compression tz0 for parallel compression”.
Without this option, tz0 is calculated by varying the “own” degree of freedom q0 and keeping “opposite” degree of freedom at zero.

Example
```
SuspF.Buf_Push.tz0 = 0.055
SuspR.Buf_Push.tz0 = 0.06 parallel
```

.Buf_Push = C*onstStiffness
.Buf_Push.1 = C*onstStiffness
.Buf_Push : *table
.Buf_Push.1 : *table

The buffer characteristic can be defined with a constant or with a data table describing the non-linear buffer behavior.
This characteristic translates buffer compression x to buffer push force. It has to be defined in first quadrant.

Syntax
Infofile table mapping with 2 columns
<compression [m]> <buffer force [N]>

Example
SuspF.Buf_Push:

0.002	32.0
0.004	88.0
0.006	167.0
0.008	269.0
0.010	393.0
0.012	596.0
0.015	1085.0

Example
SuspF.Buf_Push = 35000

*.Buf_Push.1.Mode = *SetupString*

Defines if the second buffer is parallel or series to the other buffer.
Possible values are: "Parallel", "Series"

Example
```
SuspF.Buf_Pull.1.Mode = Parallel
```

*.Buf_Pull.Amplify = *value*
*.Buf_Pull.1.Amplify = *value*

Pull buffer amplification factor.

Example
```
SuspF.Buf_Pull.Amplify = 1.0
```

*.Buf_Pull.tz0 = *value* [parallel]
*.Buf_Pull.1.tz0 = *value* [parallel]

Pull buffer mount position (defines rebound clearance). According to (EQ 57) and Figure 11.12 the pull buffer only acts for vertical displacements tz (of wheel carrier) smaller than tz0(=x_{0Pull}).
The option “parallel” means “compression tz0 for parallel compression”.
Without this option, tz0 is calculated by varying the “own” degree of freedom q0 and keeping “opposite” degree of freedom at zero.

Example
```
SuspF.Buf_Pull.tz0 = -0.09
```

.Buf_Pull = *ConstStiffness
.Buf_Pull.1 = *ConstStiffness
.Buf_Pull : *table
.Buf_Pull.1 : *table

The buffer characteristic can be defined with a constant or with a data table describing the non-linear buffer behavior.
This characteristic translates buffer compression x to buffer pull force. It has to be defined in first quadrant..

Syntax Infofile table mapping with 2 columns
<compression > <buffer force>

Example SuspF.Buf_Pull:

0.002	32.0
0.004	88.0
0.006	167.0
0.008	269.0
0.010	393.0
0.012	596.0
0.015	1085.0

Example SuspF.Buf_Pull = 35000

.Buf_Pull.1.Mode = *SetupString

Defines if the second buffer is parallel or series to the other buffer.
Possible values are: "Parallel", "Series"

Example `SuspF.Buf_Pull.1.Mode = Parallel`

11.5 Suspension Roll Stabilizer / Anti-Roll Bar

This module simulates a roll stabilizer (stabi). A force F_{Stabi} acts if there is a difference between the right t_{zr} and the left t_{zl} wheel compression. The roll stabilizer can be defined using a deflection length difference or using a deflection angle difference.

Model with deflection length difference:

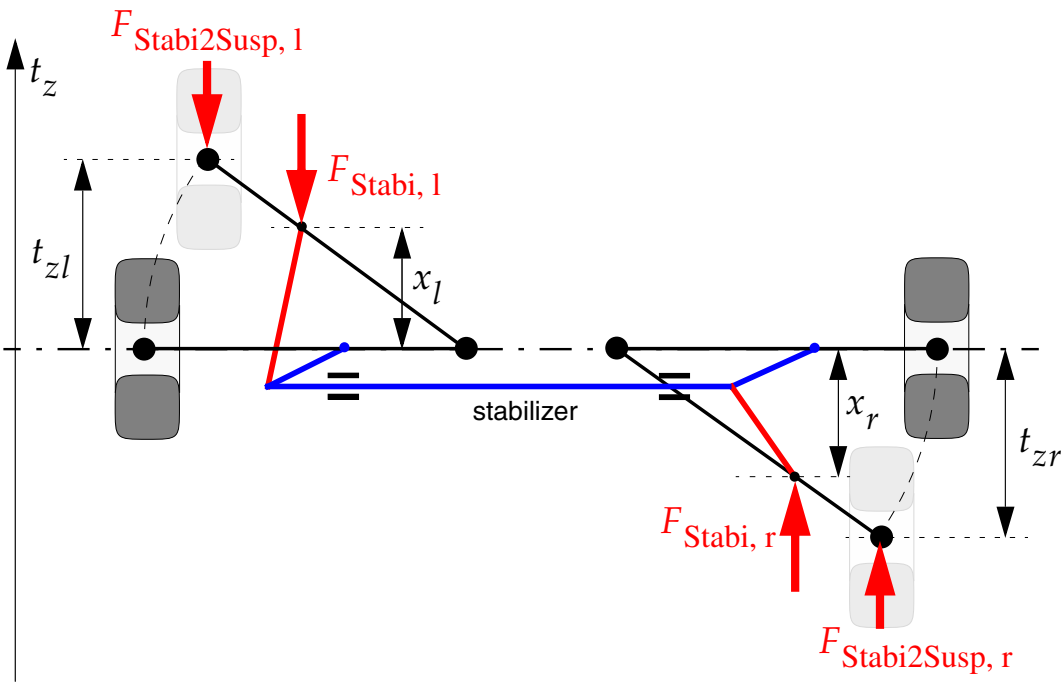


Figure 11.13: Roll stabilizer with deflection length difference

In general, according to Figure 11.13, the wheel compression ratio is not equal to the stabilizer deflection difference ($t^* \neq x^*$).

The stabilizer deflection difference x^* is defined by a deflection length difference:

$$\begin{aligned} t^* &= t_{zr} - t_{zl} \\ x^* &= x_r - x_l \end{aligned} \quad , \qquad \text{(EQ 58)}$$

The forces $F_{\text{Stabi,l,r}}$ calculate to:

$$\begin{aligned} F_{\text{Stabi, l}} &= \text{amp} \cdot f(x^*) = \text{amp} \cdot c_{\text{Stabi}} \cdot x^* \\ F_{\text{Stabi, r}} &= -F_{\text{Stabi, l}} \end{aligned} \quad , \qquad \text{(EQ 59)}$$

with the amplification factor amp and the stabilizer “spring” rate c_{Stabi}. Under normal circumstances the amplification factor should be set to 1 (only for testing and scaling).

The forces are transformed in direction of the wheel deflection as follows:

$$\begin{aligned} F_{\text{Stabi2Susp, l}} &= F_{\text{Stabi, l}} \cdot \frac{\partial x_l}{\partial q_l} \\ F_{\text{Stabi2Susp,r}} &= F_{\text{Stabi, r}} \cdot \frac{\partial x_r}{\partial q_r} \end{aligned} \quad . \qquad \text{(EQ 60)}$$

For the calculation of the stabilizer forces according to (EQ 59) the current stabilizer deflection difference x^* is needed. It is obtained from the suspension kinematics module according to Figure 11.14. q represents the generalized coordinates of the suspension.

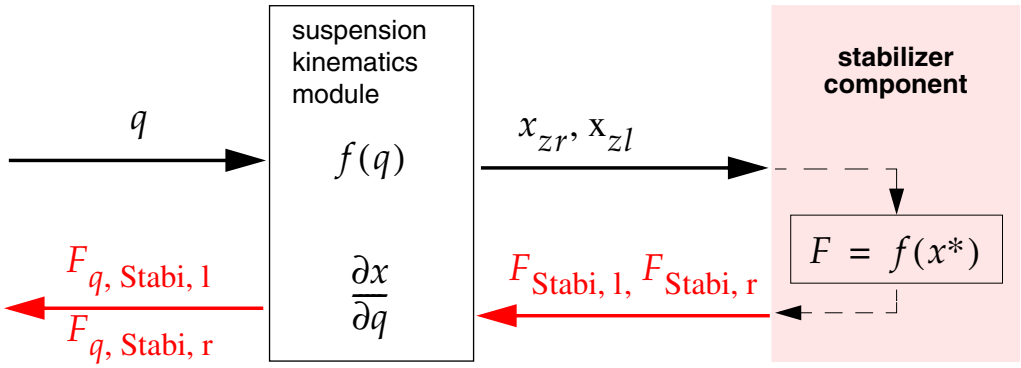


Figure 11.14: Transformation of stabilizers deflections and forces

If *no* stabi length is parametrized, the wheel compression tz is used.

Parameters

SuspF means front axle, SuspR means rear axle

SuspF.Stabi.Amplify = ***value***
SuspR.Stabi.Amplify = ***value***

Stabilizer amplification factor

```
SuspF.Stabi.Amplify = 1.0
```

.Stabi =** ***ConstStiffness
.Stabi:** ***DataTable

The stabilizer characteristic can be defined with a constant or with a data table describing the non-linear spring behavior.
Stabilizer “spring” rate c_{Stabi}. In this case the unit is N/m.

- In literature often a reciprocal compression with ${}^{\mathrm{alt}}x^* = |x_r| = |x_l|$ is used for definition of c_{Stabi}. This definition deviates to the definition of x^* in (EQ 58):

$$ {}^{\mathrm{alt}}x^* = \frac{x^*}{2}. \qquad \text{(EQ 61)} $$

  Stabilizer constants ${}^{\mathrm{alt}}c_{\mathrm{Stabi}}$ determined by this alternative definition have to be converted to the CarMaker definition of c_{Stabi}:

$$ c_{\mathrm{Stabi}} = \frac{{}^{\mathrm{alt}}c_{\mathrm{Stabi}}}{2} \qquad \text{(EQ 62)} $$

Example `SuspF.Stabi = 15000.0`

Model with deflection angle difference:

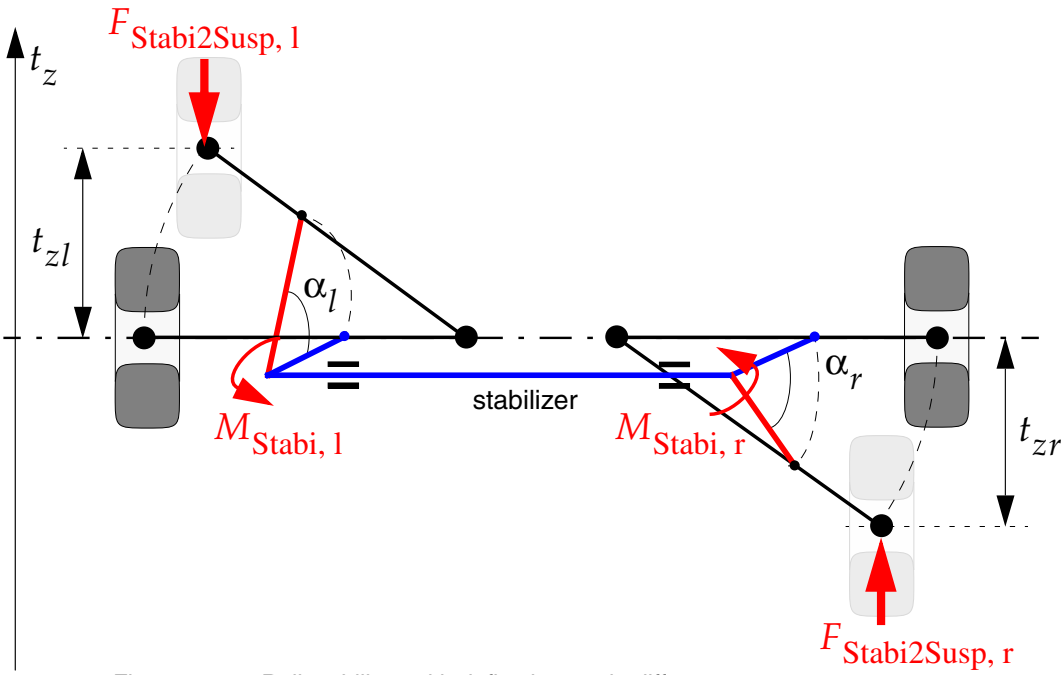


Figure 11.15: Roll stabilizer with deflection angle difference

The stabilizer deflection difference x^* is defined by a deflection angle difference:

$$t^* = t_{zr} - t_{zl}$$
$$x^* = \alpha_r - \alpha_l \qquad \text{(EQ 63)}$$

The moments $M_{\text{Stabi,l,r}}$ calculate to:

$$M_{\text{Stabi, l}} = \text{amp} \cdot f(x^*) = \text{amp} \cdot c_{\alpha\text{Stabi}} \cdot x^*$$
$$M_{\text{Stabi, r}} = -M_{\text{Stabi, l}} \quad , \qquad \text{(EQ 64)}$$

with the amplification factor amp and the stabilizer "torsion spring" rate $c_{\alpha\text{Stabi}}$.

The moments are transformed in direction of the wheel deflection as follows:

$$F_{\text{Stabi2Susp, l}} = M_{\text{Stabi, l}} \cdot \frac{\partial \alpha_l}{\partial q_l}$$
$$F_{\text{Stabi2Susp,r}} = M_{\text{Stabi, r}} \cdot \frac{\partial \alpha_r}{\partial q_r} \qquad \text{(EQ 65)}$$

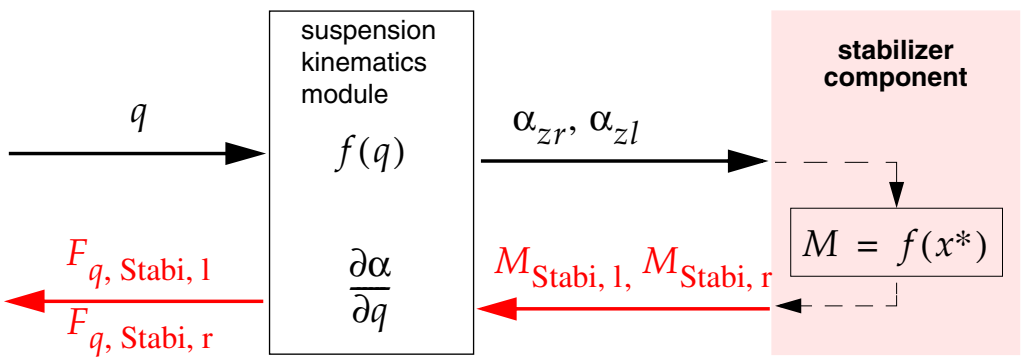


Figure 11.16: Transformation of stabilizers deflections and moments

11.6 Wheel Bearing Friction

CarMakes gives the possibility to calculate the friction torque in the wheel bearing. For the calculation of the friction torque a simplified relationship is used:

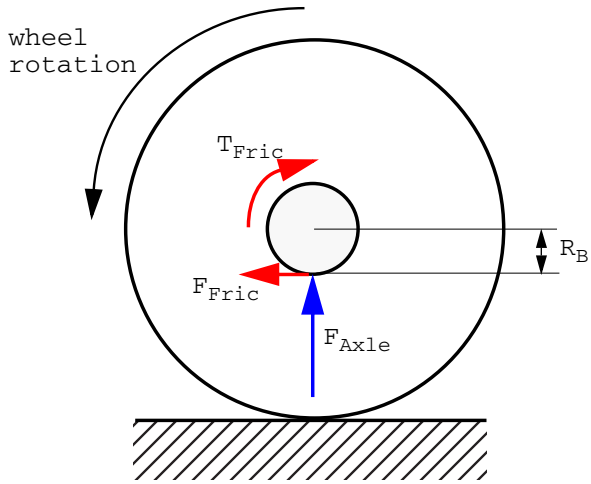


Figure 11.17: Friction torque in the wheel bearing

$$T_{\mathrm{Fric}} = \mathrm{amp} \cdot F_{Fric} \cdot R_B = \mathrm{amp} \cdot F_{Axle} \cdot \mu_{Fric} \cdot R_B \qquad \textbf{(EQ 66)}$$

Parameters

.WhlBearing.On = *bool

Parameter for the activation of friction torque calculation. Default: 0.

.WhlBearing.Amplify = *Factor

Amplification factor of the wheel bearing friction torque ‘amp’. Default: 1.0.

.WhlBearing.Coeff = *FrictionCoefficient

Friction coefficient of the wheel bearing μ_{Fric}. Default: 0.003.

.WhlBearing.Radius = *BearingRadius

Indicates the radius of the wheel bearing R_B. Unit: [m]. Default: 0.03.

Chapter 12

Suspension Kinematics and Compliance

12.1 Overview

Kinematics describes the spacial movements of a wheel due to compression and steer action. Two cases of kinematics are distinguished:

- *Suspension kinematics* (due to pure wheel compression).
- *Steering kinematics* (due to pure steer actions).

In reality a superposition of those two isolated cases exists.

Compliance describes the spacial movements of a wheel due to wheel forces which cause elastic deformations of the wheel suspension. Because of the complex construction of a vehicle suspension forces and torques can produce movements of the wheel in other directions than in their effective direction.

Movements are described through coordinates in an axis system. The center of this axis system is the wheel center. Coordinates for translations and rotations of this axis system are used.

Primary and Secondary Coordinates of the Suspension

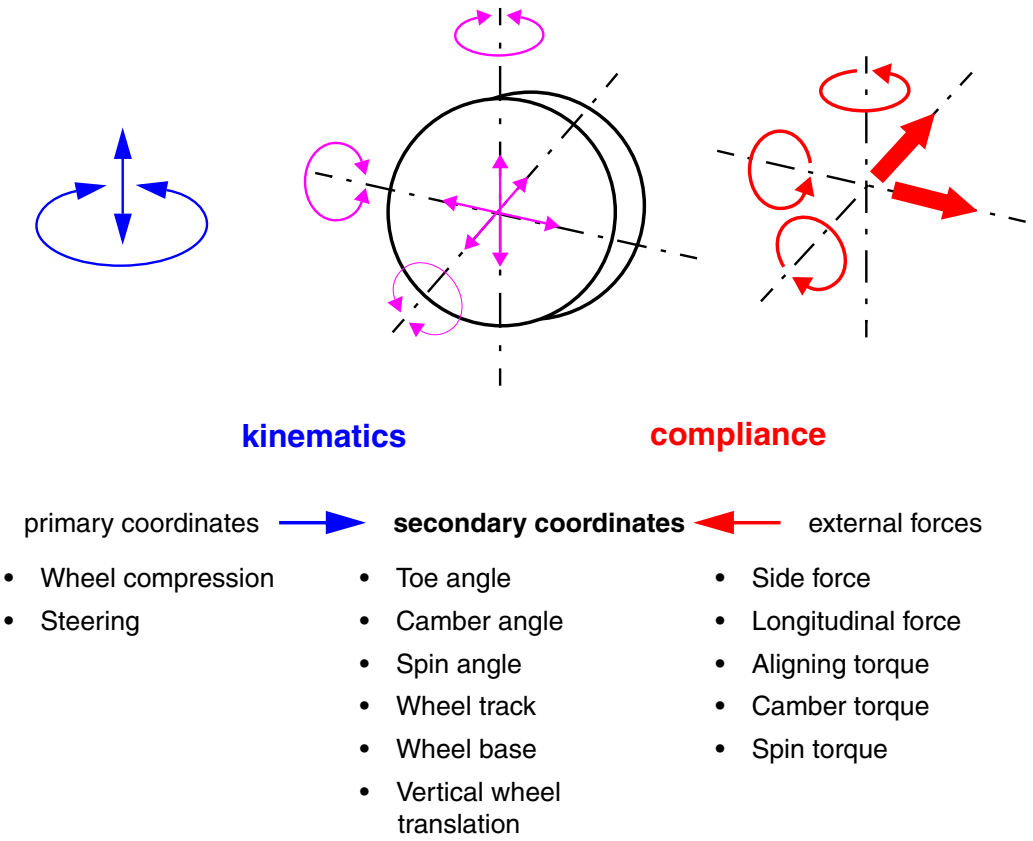


Figure 12.1: Kinematics and Compliance

Figure 12.1 shows the context of kinematics and compliance. Wheel compression and steer actions are called *primary coordinates* of a wheel. A change in the primary coordinates has an effect on the *secondary coordinates*. Kinematics is defined as force free movements of the wheel suspension and is measured as a function of the primary coordinates.

Contrary to the kinematics the compliance investigates in change of secondary coordinates as a result of external forces act upon the wheel. External forces (and torques) can arise from the wheel contact point the brakes and the power transmission. They are transferred to the wheel carrier. These forces yield to changes of the secondary coordinates of the wheel because of elasticities of the suspension.

To describe the position of the wheel usually terms from the vehicle dynamics vernacular are used (like in the middle section of Figure 12.1). For parametrization of wheel suspensions CarMaker uses more general, simulation technical terms.

The spacial movements of a wheel carrier in CarMaker is equal to the movements of the wheel carrier axis system `Fr2` (see section 1.2 'CarMaker Axis Systems').

The following figures explain the relations of the different viewpoints and their conversions:

Toe Angle / Rotation r_z:

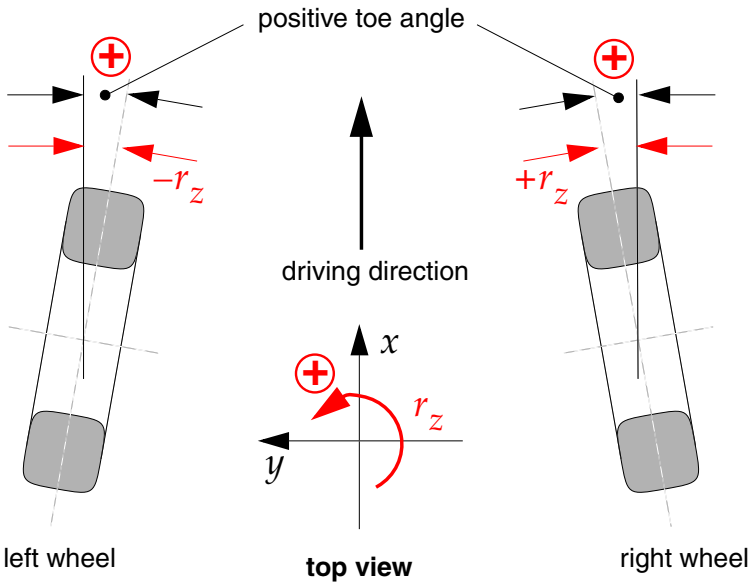


Figure 12.2: Definition of the tow angle and rotation r_z

A *positive toe angle* at the *right wheel carrier* equals a *positive rotation* r_z of the Fr2 axis system.

Camber Angle / Rotation r_x:

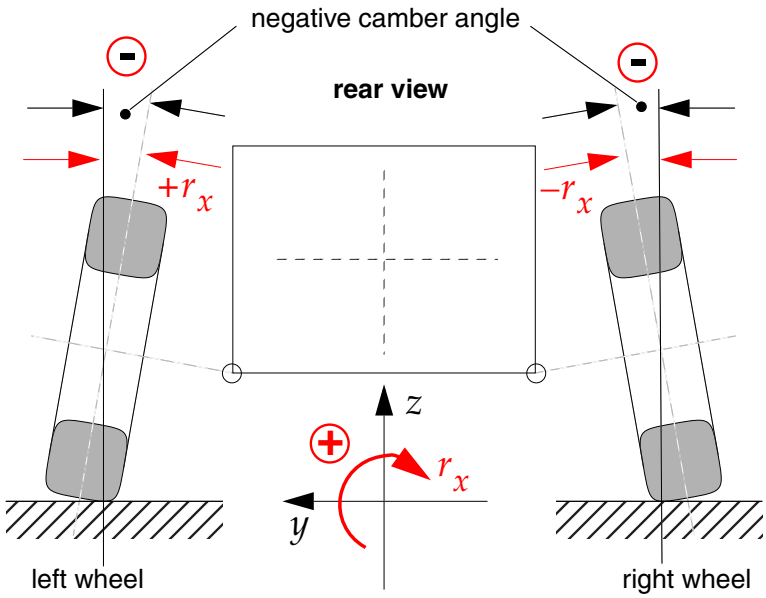


Figure 12.3: Definition of camber angle and rotation r_x

A *positive camber angle* at the *right wheel carrier* equals a *positive rotation* r_x of the Fr2 axis system.

Spin Angle / Rotation r_y:

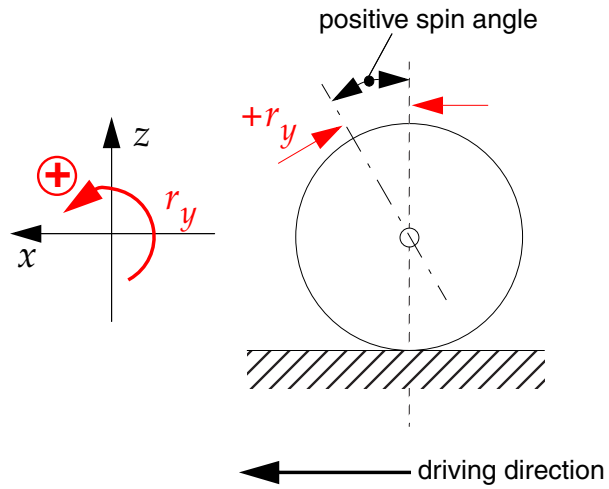


Figure 12.4: Definition of spin angle rotation r_y

A *positive spin angle* equals a *positive rotation* r_y of the Fr2 axis system.

Translation t_y:

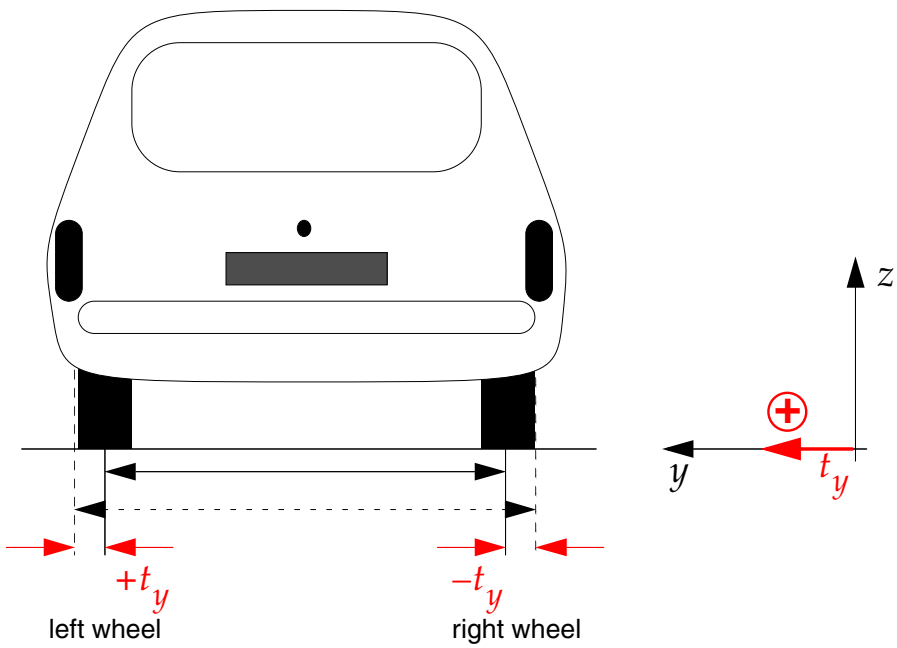


Figure 12.5: Definition of wheel track and translation t_y:

The translations t_y of left and right wheel are parametrized independently. The change of wheel track is a result of both, the change of the left and the right wheel.

Translation t_x:

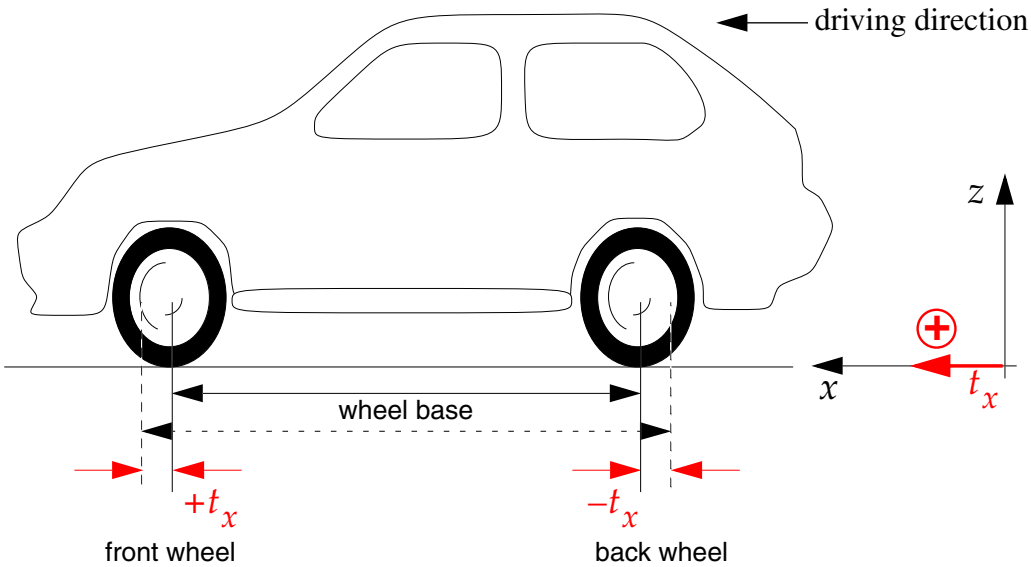


Figure 12.6: Definition of wheel base and translation t_x

The translations t_x of front and back wheel are parametrized independently. The change of wheel base is a result of both, the change of the front and the back wheel.

Wheel compression / Translation t_z:

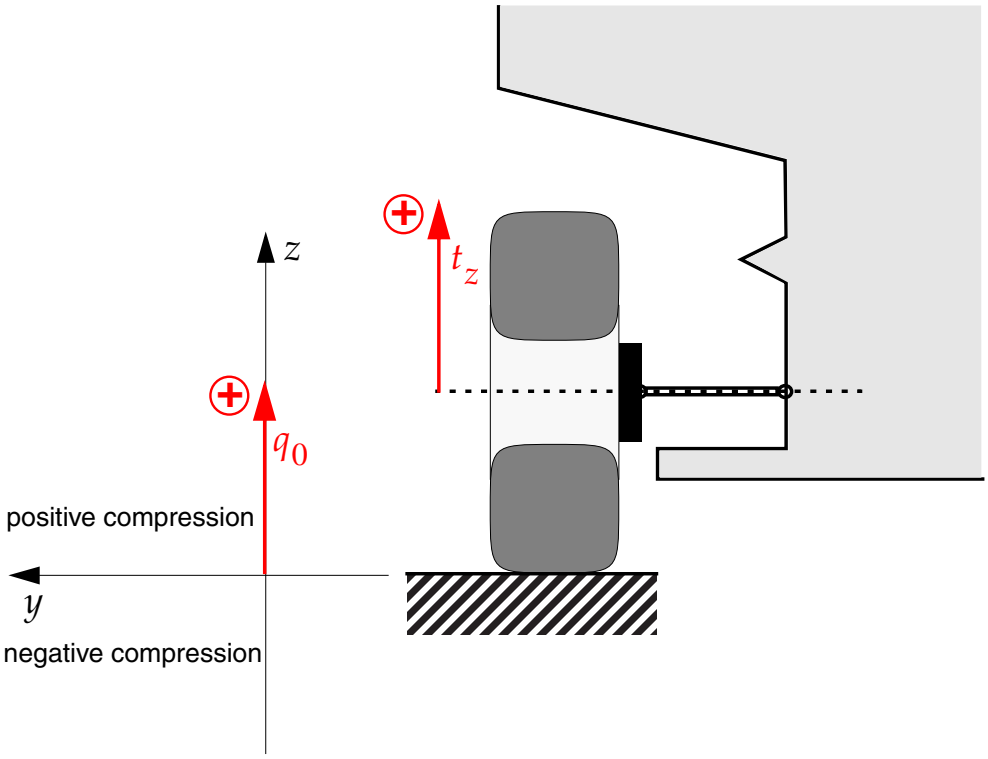


Figure 12.7: Definition of wheel compression and translation t_z

Positive wheel compression q_0 is defined in upward direction. Usually translation $q_0 = 0$ equals the wheel position in vehicles design configuration.

12.1.1 Describing Kinematics with Generalized Coordinates

CarMaker uses *generalized coordinates* to describe kinematic movements.

Basically generalized coordinates q_i are independent coordinates to describe a system with i degrees of freedom.

$$i = 6N - k \qquad \textbf{(EQ 67)}$$

N	number of bodies
k	number of holonomic constraints

Requirements for generalized coordinates are:

- All states of a system can be described with one set of generalized coordinates respecting all holonomic constraints.
- No holonomic constraints exist for generalized coordinates.

The last statement especially means that one coordinate can have any value without affecting the value of another coordinate.

Applied to the kinematics e. g. of a front axle this means that usually two or three generalized coordinates exist to describe the kinematic movements of this axle. This is already implied in Figure 12.1 as:

- A generalized coordinate q_0 to describe the degree of freedom for wheel compression.
- A generalized coordinate q_1 to describe the degree of freedom for opposite wheel compression (used for a front rigid axle).
- A generalized coordinate q_2 to describe the degree of freedom for the steer influence.

CarMaker gives *no* restrictions for the choice of the generalized coordinates q_0, q_1, q_2. In other words the user is responsible to choose generalized coordinates that fulfil the requirements mentioned above.

Conceptual Overview of Kinematics Calculation

This section describes the internal procedure how CarMaker calculates kinematic movements with the user defined kinematic characteristics.

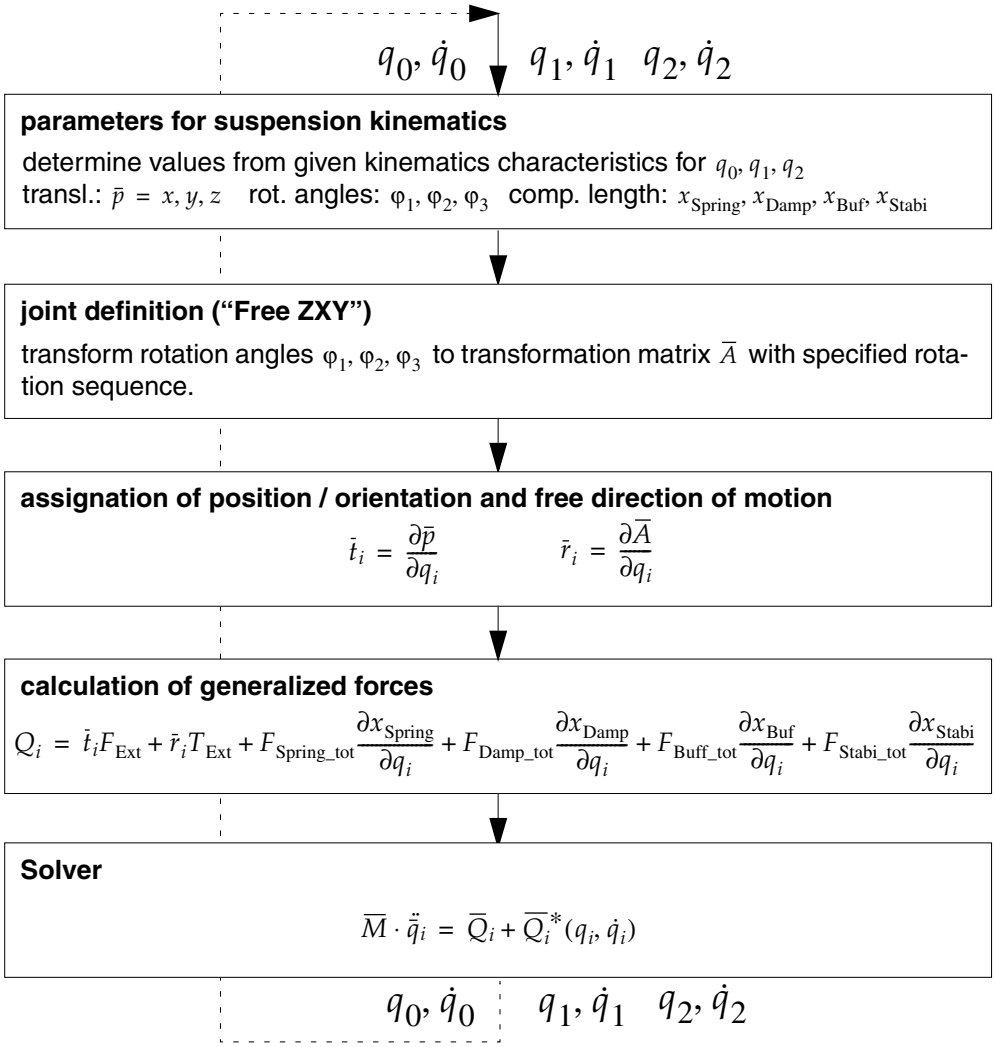


Figure 12.8: Kinematics calculation

The first box in Figure 12.8 shows the quantities acquired from the user defined characteristics for the given generalized coordinates q_0, q_1, q_2. The vector $\bar{p}$ represents the three *translation offsets* and the rotation angles $\varphi_1, \varphi_2, \varphi_3$ specify three *cardan angles*. Additionally the *component lengths* of spring, damper, buffers and stabilizer are acquired.

With the second step the cardan angles $\varphi_1, \varphi_2, \varphi_3$ are transformed by the joint definition to the *transformation matrix* $\bar{A}$. A specific rotation sequence (e. g. 'ZXY') is provoked by the joint definition.

For calculation of the generalized forces the free directions of motion in directions of q_0, q_1, q_2 are needed. This is done with the third step by partial differentiation of $\bar{p}$ and $\bar{A}$.

In the fourth step for each generalized coordinate a *generalized force* is computed. External forces and torques are respected in direction of the free direction of motion $\bar{t}_i$ and $\bar{r}_i$ for each coordinate. External forces and torques result from *tires*, *brakes*, *gravity*, etc. Also the total (internal and external) forces of spring, damper, buffers and stabilizers are respected pro rata in the direction of the generalized coordinate.

As a final step the solver computes and integrates the generalized accelerations $\ddot{q}_0, \ddot{q}_1, \ddot{q}_2$. Beside the generalized forces $\overline{Q}_i$ pseudo forces $\overline{Q}_i^*$ resulting from coriolis forces are taken into account. These pseudo forces $\overline{Q}_i^*$ depend form $q_i, \dot{q}_i$ instead from $\ddot{q}_i$.

Choosing Generalized Coordinates

After dealing with the basics of kinematics definitions in CarMaker this section tries to give practical hints about how to choose appropriate generalized coordinates.

Front Axle

Two cases are distinguished for a front axle:

- Independent front axle: For a front axle a generalized coordinate q_0 is needed to describe the degree of freedom for wheel compression and a generalized coordinate q_2 is needed to describe the degree of freedom for the steer influence. A good recommendation is:

Generalized Coordinate	Description
q_0	Wheel compression $t_{z,0}$ for steer angle zero (driving direction straight forward usually equals to $q_2 = 0$).
q_2	The rack displacement.

The reason why q_0 is chosen as $t_{z,0}$ for steer angle zero is based on the fact that the steer influence changes the compression of the two controlled wheels. This effect is called self-alignment due to lift of the suspension subframe. Such a definition for q_0 is necessary because of the self-alignment the wheel compression t_z depends from both coordinates q_0 and q_2. With the definition above the requirements for generalized coordinates are fulfilled. Both coordinates are independent.

- Dependent front axle: For a dependent front axle additionally a generalized coordinate q_1 is needed to describe the degree of freedom for the opposite wheel compression:

Generalized Coordinate	Description
q_0	Wheel compression t_z.
q_1	Wheel compression t^*_z of the opposite wheel.
q_2	The rack displacement.

Rear Axle

Two cases are distinguished for a rear axle:

- Dependent rear axle: The orientation of one wheel *also* depends on the compression of the other wheel (e. g. twist-beam rear axle). Good choice:

Generalized Coordinate	Description
q_0	Wheel compression t_z.
q_1	Wheel compression t^*_z of the opposite wheel.

- Independent rear axle: The orientation of one wheel *only* depends on the compression of this wheel. (e. g. independent rear suspension). Good choice:

Generalized Coordinate	Description
q_0	Wheel compression t_z.

12.1.2 Brief Introduction to the Measurement Procedure of K&C parameters

In practice often K&C test rigs or likewise simulations are used to measure the required K&C parameters. This chapter gives a brief introduction how to measure a vehicle to obtain data that suits to the CarMaker K&C data files.

The following table shows terms and abbreviations used to describe the measurement procedure.

Name	Definition
Reference System	As reference system a axis system similar to the CarMaker Fr0 axis system is used (See section 1.2). The Vehicle is symmetrically aligned, nose pointing to increasing values of the X-axis.
Static Equilibrium Configuration (**SEC**)	Means equilibrium position of the (empty) vehicle on even ground. The body is not fixed relative to the reference frame. The suspension travel *must* be symmetrical (left/right). The SEC compression can be measured.
Wheel Travel	Translation of wheel center in Z-direction.
Compression Travel	Travel from SEC to max. compression (wheel carrier near to body) = up to metal-to metal position, in general
Rebound Travel	Travel from SEC to max. decompression (wheel carrier far away from body)
Wheel Load (**Fz**)	Vertical force acting from environment onto wheel carrier, positive if directed upwards. Forces are applied (point of attack) to the wheel center or the tyre/road contact point (=tire patch).
SWA	Steering wheel angle
Deflection Procedure (**DP**)	The expression *deflection procedure* is used synonymously for the following test sequence: • Start from SEC. • Slowly increase Fz (= compress suspension) up to wheel load max. value (e.g. 10000 N) = compression travel. • Slowly decrease Fz (=decompress suspension, rebound) up to wheel load min value (e.g. 100 N) = rebound travel. • Go back to SEC. • Compression and rebound speed should be slow and identical in both direction (thereby eliminating velocity depended effects) • Front axle: Keep steering rack fixed (eliminate them, if possible effects coming from steering compliance, steering backlash, ...). • No horizontal (Fx, Fy) force (ground to tyre) is transmitted. • No torque (Mx, My, Mz) (ground to tyre) is transmitted. • Wheels are rigidly fixed to wheel carrier (braking conditions).
Wheel Replacement System	Using a set of artificial wheels replacing real wheels during elastokinematics measurements. Advantage: Higher forces (Fy, Fz) during elastokinematic measurements (no slip). Measurements for the elastokinematics are therefore preferably done with wheel replacement system

The following lists describes two procedures to be accomplished to acquire the required data for the CarMaker K&C characteristics:

Kinematics measurement procedure

See also section 12.3.2 for information on how to parametrise SKC-files.

- Find static equilibrium configuration (SEC)
- Determine absolute kinematic quantities (i.e. relative to `Fr0`) in SEC.
- Rigidly fix vehicle body in SEC relative to `Fr0`. Vehicle body remains clamped and restrained during K&C measurements.
- For *Front* axles:
  - Do DP (left & right *symmetrical*) keeping steering rack fixed, SWA = 0 (straightforward driving)
  - Do DP (left & right *asymmetrical*), 50mm compression and rebound are sufficient, SWA = 0 (straightforward driving)
  - Do DP (left & right symmetrical) while changing SWA = min value, -200, -100, -60, -40, -20, -10, 0, 10, 20, 40, 60, 100, 200, max value [deg]. In order to increase the model quality: A finer SWA-graduation is possible/useful.
  - Determine *steering reduction ratio*: Fix (relative to `Fr0`) Wheel Carrier in SEC. Starting from min value, increase SWA in steps of 5[deg] (around zero position) and 10 [deg] (for steering wheel angles > 30[deg]) up to max. value (lock to lock steering). Measure kinematic quantities.
- For *independent rear* axles: Do DP (left & right symmetrical)
- For *Twist-beam rear* axles (dependent rear axle):
  - Fix rear right wheel carrier relative to `Fr0` in SEC, do DP with rear-left wheel carrier
  - Increase (up to max. compression) in steps of 5-25mm rear right wheel carrier and fix it (relative to `Fr0`), do DP with rear left wheel carrier
  - Decrease (up to max rebound) in steps of 5-25 mm rear right wheel carrier and fix it (relative to `Fr0`), do DP with rear left wheel carrier

Elastokinematics measurement procedure (front & rear)

- Fix vehicle body in SEC.
- Block wheels (wheel carrier rigidly fixed to wheel)
- Longitudinal compliance (longitudinal acceleration and Braking compliance)
  - Apply horizontal forces Fx = -6000N, - 5000N, -4000N, ... , 0N, 1000 N, ... , 6000 N at front left and rear left wheel
  - Measure kinematic quantities
  - Forces are applied at wheel center or at ground-tyre contact point.
- Lateral compliance:
  - Apply horizontal forces Fy = -6000N, - 5000N, -4000N, ... , 0N, 1000 N, ... , 6000 N at front left and rear left wheel
  - Measure kinematic quantities
  - Forces are applied at wheel center or at ground/tyre contact point.
- Aligning torque compliance
  - Apply vertical torque Mz = -200 Nm, -100 Nm, 0Nm, 100 Nm, 200 Nm at front left and rear left wheel
  - Measure kinematic quantities
  - Torque is applied at wheel center or at ground/tyre contact point.
- In-phase / out-of-phase loading (for compliance model Displace2D)
  - Apply forces/torques at front left/right wheels and rear left/right wheels
  - Measure kinematic quantities

It is recommended to apply the tire forces and torque while the measurement procedure at wheel center, because CarMaker expects in the K&C data file for the elastokinematics the measured forces/torques and the corresponding wheel travel in wheel center.

12.2 Kinematics and Compliance

The suspension is connected to the vehicle body by a translational and rotational kinematic joint (rotation sequence: Z–X–Y).

$$\begin{bmatrix} t \\ r \end{bmatrix} = \sum_i Kin(q_0, q_1, q_2) + \sum_j Com(q_0, q_1, q_2, F, \ldots) \qquad \text{(EQ 68)}$$

The suspension kinematics and compliance can be defined by a number of models or parameter sets, which are calculated in the order of there definition.

An additional outward shift of the wheel can be defined in the tire parameter set, see section 17.2 'General Tire Parameters'.

Parameters

The following parameters are required for suspension kinematics and suspension compliance to parametrise the K&C data in a SKC-File:

All model parameters have a prefix, depending on the corresponding suspension and the number of the model definition. The prefix can be an empty string or a string, ending with a dot ("SuspF.", "SuspF.Kin.3." for kinematics, "SuspR.Com.2." for compliance).

SuspKey.KnC.*N =** ***N

The kinematics or compliance definition consists of *N* superimposed models, see (EQ 68). The suspension is selected by *SuspKey* which can be **SuspF** or **SuspR**. The *string KnC* is **Kin** for kinematics or **Com** for compliance.

It is an optional entry, the default is **1**.

SuspKey.KnC.i.*FName =** ***FileName **or**
SuspKey.KnC.i.*Kind =** ***ModelKind

The i-th model is selected by the Kind entry *ModelKind.* Instead of using the Kind key, an external parameter set can be referenced by filename *FileName*. The reference to an external parmeter set is supported only on toplevel and in case of only one model at toplevel, not in a parameter set referenced itself. *FileName* is the path relative to the suspension kinematics directory.

SuspKey.KnC.i.*ValidSide =** ***ValidSide

The i-th model is valid for *ValidSide*. If suspension side and *ValidSide* doesn't match, this model is skipped for this suspension side.

ValidSide is always **left+right**. It's an optional entry, the default is left+right.

SuspKey.KnC.i.*InputSide = *InputSide

The i-th model is defined by parameters for the suspension on side *ValidSide*.

The model parameters for the current suspension side has to be deduced from the input parameters for side *InputSide*.

InputSide is exactly one of **left**, **left+right** and **right**. It is an optional entry, the default is **left**.

SuspKey.KnC.i.*RotOrder = *RotOrder

RotOrder is exactly one of **zxy** and **zyx**. It's an optional entry, the default is **zxy**. It defines the rotation order in which the angles in the SKC-File are given.

***SuspKey.KnC.i.*L.*param* = ...**
***SuspKey.KnC.i.*R.*param* = ...**

Parameters for the left suspension starts with "L.", for the right with "R.".

Example

```
SuspF.Kin.N =                1
SuspF.Kin.0.Kind =           MapNL
SuspF.Kin.0.ValidSide =      left+right
SuspF.Kin.0.InputSide =      left+right
SuspF.Kin.0.RotOrder =       zxy
...
```

12.3 Kinematics Models

All model parameters have a prefix, depending on the corresponding suspension and the number of the model definition. The prefix can be an empty string or a string, ending with a dot ("SuspF.", "SuspF.Kin.3." for kinematics, "SuspR.Com.2." for compliance).

12.3.1 “Linear 1 DOF”, “Linear 2 DOF” and “Linear 3 DOF”

Description

Linear kinematics uses linear equations to describe the kinematic movements for translations and rotations of the wheel (-carrier). Depending on the number of generalized coordinates the model *Linear 1 DOF* is used for one , the model *Linear 2 DOF* is used for two and the model *Linear 3 DOF* is used for three generalized coordinates.

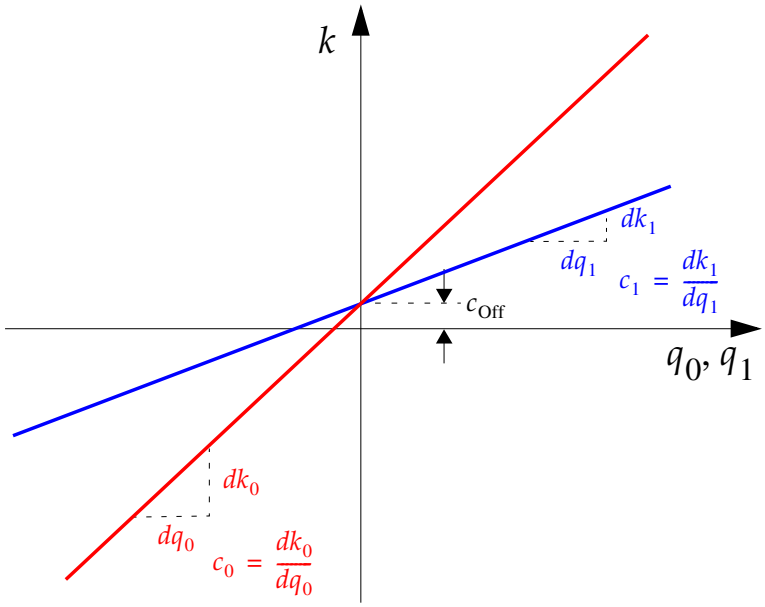


Figure 12.9: Kinematics ‘linear2D’ for coordinates q_0, q_1

This means (steered) independent front axles use the model *Linear 2 DOF* because they have two degrees of freedom which influence the kinematics of the wheel carrier.

For instance, dependent front axles has to use the model *Linear 3 DOF*, where kinematics depends on three degrees of freedom.

Independent rear suspensions use the model *Linear 1 DOF* because the kinematics only depends on the wheel compression of the considered wheel carrier.

For instance, for a twist beam rear axle, where kinematics also depends on the compression of the opposite wheel carrier, the model *Linear 2 DOF* has to be used.

A simple linear equation is used for each coordinate.

$$k(q_0) = c_{Off} + c_0 \cdot q_0 \qquad \text{(EQ 69)}$$

The model ‘Linear 2 DOF’ uses a superposition of two linear equations one for each generalized coordinate.

$$k(q_0, q_1) = c_{Off} + c_0 \cdot q_0 + c_1 \cdot q_1 \qquad \text{(EQ 70)}$$

$$k(q_0, q_2) = c_{Off} + c_0 \cdot q_0 + c_2 \cdot q_2 \qquad \text{(EQ 71)}$$

The model *Linear 3 DOF* uses a suporposition of three linear equations one for each generalized coordinate.

$$k(q_0, q_1, q_2) = c_{\mathrm{Off}} + c_0 \cdot q_0 + c_1 \cdot q_1 + c_2 \cdot q_2 \qquad \textbf{(EQ 72)}$$

with:

k	one of 3 translations (t_x, t_y, t_z) in frame Fr1 or 3 cardan rotation angles (r_x, r_y, r_z) with a rotation sequence Z–X–Y or 4 component lengths ($l_{\mathrm{Spring}}, l_{\mathrm{Damp}}, l_{\mathrm{Buf}}, l_{\mathrm{Stabi}}$)
c_{Off}	offset for $q_0, q_1, q_2 = 0$
c_0	gradient depending on compression q_0
c_1	gradient depending on compression q_1 of opposite wheel
c_2	gradient depending on steering coordinate q_2

Parameters

The following parameters are required for this model. The asterisk "*" is an abbreviation for the suspension, for kinematics or compliance, for the model number and the parameter side, see section 12.2 'Kinematics and Compliance' on page 145.

The corresponding *ModelKind* for the kinematic models are:

ModelName	ModelKind	Description
Linear 1 DOF	Linear	Kinematic depends on one degree of freedom
Linear 2 DOF	Linear2D	Kinematic depends on two degrees of freedom
Linear 3 DOF	Linear3D	Kinematic depends on three degrees of freedom

Linear 1 DOF:

***.tx =**	***tx0***	***dtx/dq0***
***.ty =**	***ty0***	***dty/dq0***
***.tz =**	***tz0***	***dtz/dq0***

Optional. Wheel (-carrier) translations. Translation is calculated by the offset and a ratio depending on the generalized coordinate q0.
Defaults: 0 0 and for tz 0 1. Units: m m/q0.

***.rx =**	***rx0***	***drx/dq0***
***.ry =**	***ry0***	***dry/dq0***
***.rz =**	***rz0***	***drz/dq0***

Wheel (-carrier) rotations.
Rotation depends on the generalized coordinate q0. Defaults: 0 0. Units: rad rad/q0.

***.lSpring =**	***lSpring0***	***dlSpring/dq0***
***.lDamp =**	***lDamp0***	***dlDamp/dq0***
***.lBuf =**	***lBuf0***	***dlBuf/dq0***
***.lStabi =**	***lStabi0***	***dlStabi/dq0***

Absolute length of the force elements spring, damper, buffer and roll stabilizer depending on the generalized coordinate q0.
Remark: For the roll stabilizer the units of this coordinate and the units of the roll stiffness has to fit together. Defaults: 0 -1. Units: m m/q0.

Linear 2 DOF:

***.tx =**	***tx0***	***dtx/dq0***	***dtx/dq(1 or 2)***
***.ty =**	***ty0***	***dty/dq0***	***dty/dq(1 or 2)***
***.tz =**	***tz0***	***dtz/dq0***	***dtz/dq(1 or 2)***

Optional. Wheel (-carrier) translations. Translation is calculated by the offset and two ratios dependending on the generalized coordinates of the suspensions (see (EQ 70),(EQ 71)). Defaults: 0 0 0 and for tz 0 1 0. Units: m m/q0 m/q(1 or 2).

***.rx =**	***rx0***	***drx/dq0***	***drx/dq(1 or 2)***
***.ry =**	***ry0***	***dry/dq0***	***dry/dq(1 or 2)***
***.rz =**	***rz0***	***drz/dq0***	***drz/dq(1 or 2)***

Wheel (-carrier) rotations. Rotation depends on the generalized coordinates of the suspension (see (EQ 70),(EQ 71)). Defaults: 0 0 0. Units: rad rad/q0 rad/q(1 or 2).

***.lSpring =**	***lSpring0***	***dlSpring/dq0***	***dlSpring/dq(1 or 2)***
***.lDamp =**	***lDamp0***	***dlDamp/dq0***	***dlDamp/dq(1 or 2)***
***.lBuf =**	***lBuf0***	***dlBuf/dq0***	***dlBuf/dq(1 or 2)***
***.lStabi =**	***lStabi0***	***dlStabi/dq0***	***dlStabi/dq(1 or 2)***

Absolute length of the force elements spring, damper, buffer and roll stabilizer depending on generalized coordinates. Defaults: 0 -1 0. Units: m m/q0 m/q(1 or 2).
Remark: For the roll stabilizer the units of this coordinate and the units of the roll stiffness has to fit together.

Linear 3 DOF:

***.tx =**	***tx0***	***dtx/dq0***	***dtx/dq1***	***dtx/dq2***
***.ty =**	***ty0***	***dty/dq0***	***dty/dq1***	***dty/dq2***
***.tz =**	***tz0***	***dtz/dq0***	***dtz/dq1***	***dtz/dq2***

Optional. Wheel (-carrier) translations. Translation depends on the generalized coordinates (see (EQ 72)). Defaults: 0 0 0 0 and for tz 0 1 0 0. Units: m m/q0 m/q1 m/q2.

***.rx =**	***rx0***	***drx/dq0***	***drx/dq1***	***drx/dq2***
***.ry =**	***ry0***	***dry/dq0***	***dry/dq1***	***dry/dq2***
***.rz =**	***rz0***	***drz/dq0***	***drz/dq1***	***drz/dq2***

Wheel (-carrier) rotations. Rotation depends on the generalized coordinates of the suspension (see (EQ 72)). Defaults: 0 0 0 0. Units: rad rad/q0 rad/q1 rad/q2.

***.lSpring =**	***lSpring0***	***dlSpring/dq0***	***dlSpring/dq1***	***dlSpring/dq2***
***.lDamp =**	***lDamp0***	***dlDamp/dq0***	***dlDamp/dq1***	***dlDamp/dq2***
***.lBuf =**	***lBuf0***	***dlBuf/dq0***	***dlBuf/dq1***	***dlBuf/dq2***
***.lStabi =**	***lStabi0***	***dlStabi/dq0***	***dlStabi/dq1***	***dlStabi/dq2***

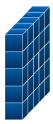
Absolute length of the force elements spring, damper, buffer and roll stabilizer depending on generalized coordinates. Defaults: 0 -1 0 0. Units: m m/q0 m/q1 m/q2.
Remark: For the roll stabilizer the units of this coordinate and the units of the roll stiffness has to fit together.

12.3.2 “MapNL”

Description

Nonlinear 1-dimensional (1D) or 2-dimensional (2D) mapping, depending on the following suspension degrees of freedom (generalised coordinates): wheel compression q_0, opposite wheel compression q_1 and steering rack displacement q_2. The kinematic values (translations, rotations of the wheel) are described as nonlinear functions (nlf) of these three generalised coordinates:

- 1D: $\mathbf{Kin} = \mathbf{nlf}(q_0)$



- 2D: $\mathbf{Kin} = \mathbf{nlf}(q_0, q_1)$ or $\mathbf{Kin} = \mathbf{nlf}(q_0, q_2)$

There are various other models, which use superposition and combine 1D and/or 2D mapings.

- 1D + 1D: $\mathbf{Kin} = \mathbf{nlf}(q_0) \oplus \mathbf{nlf}(q_1)$ or $\mathbf{Kin} = \mathbf{nlf}(q_0) \oplus \mathbf{nlf}(q_2)$

- 1D + 1D + 1D: $\mathbf{Kin} = \mathbf{nlf}(q_0) \oplus \mathbf{nlf}(q_1) \oplus \mathbf{nlf}(q_2)$

- 2D + 1D: $\mathbf{Kin} = \mathbf{nlf}(q_0, q_1) \oplus \mathbf{nlf}(q_2)$ or $\mathbf{Kin} = \mathbf{nlf}(q_0, q_2) \oplus \mathbf{nlf}(q_1)$

- 2D + 2D: $\mathbf{Kin} = \mathbf{nlf}(q_0, q_2) \oplus \mathbf{nlf}(q_0, q_1)$

High fidelity MapNL models use lookup-tables to describe the spacial movements of the wheel.

Location and Activation

The K&C parameter files are stored under `<ProjectDirectory>/Template/Data/Chassis`. The high fidelity MapNL models are activated by choosing the relevant SKC-file from within the kinematics tab of the Vehicle Data Set editor in CarMaker. The parametrised tables will then be used.

Units

All quantities, if not explicitly mentioned otherwise are SI-Units. Particularly the following quantities are used in the K&C parameter files.

Table 12.1: SI Units used with CarMaker High Fidelity K&C Parameters

Quantity	Name	Symbol
Time	second	s
Length	meter	m
Angle	radian (one turn = 2*p)	rad
Mass	kilogram	kg
Force	newton	N
Torque	newton meter	Nm
Stiffness	newton per meter	N/m
Rotational Stiffness	newton meter per radian	Nm/rad

Table Concept

The characteristics for high fidelity K&C are stored in a specific indexed table format. Characteristics can be one or two dimensional depending on the number of generalised coordinates. The vectors Arg_0 and optional Arg_1, Arg_2 and Arg_3 hold the values of all sample points for the lookup tables. The values corresponding to the sample points are specified in a list.

Parameters

The following parameters have to be specified in a SKC-file:

***.Kind =**	***MapNL***
***.ValidSide =**	***left+right***
***.InputSide =**	***left (or left+right)***

In case you only parametrise tables for the left side, these values are then mirrored to the right side by CarMaker.

***.Arg =**	***keyword 1*** **<keyword 2> <keyword 3>**

The following table shows the possible keyword combinations, which specify the model to be used:

Table 12.2: Keyword combinations and other labels for all models

Model	Keyword combination	Arguments	Mappings
1D	comp	Arg0	Data
1D + 1D	comp comp comp steer *(see below for information on compatibility)*	Arg0, Arg1 Arg0, Arg1	Data, Data1 Data, Data1
1D + 1D + 1D	comp comp steer *(see below for information on compatibility)*	Arg0, Arg1, Arg2	Data, Data1, Data2
2D	compsteer compcomp	Arg0, Arg1 Arg0, Arg1	Data Data
2D + 1D	compcomp steer compsteer comp	Arg0, Arg1, Arg2 Arg0, Arg1, Arg2	Data, Data1 Data, Data1
2D + 2D	compsteer compcomp	Arg0, Arg1, Arg2, Arg3	Data, Data1

The first keyword must always be "comp" or start with “comp”. Two keywords written together without a blank indicate a 2D mapping. Keywords separated by at least one blank indicate the superposition of two (or three) mappings.

To use the new models, `FileIdent` has to be set to `CarMaker-SuspKnC-* 2`. Old SKC-files can still be used. CarMaker will check `FileIdent` and in case `CarMaker-SuspKnC-* 1` is set, CarMaker will transform the old keyword combination to the corresponding new keyword combination. So nothing in the old files has to be changed.

.Arg0 =** ***min ... max
.Arg0.Fac2SI =** ***Fac2SI

Values of the first kinematics generalised coordinate. There should be at least five sample points.The number of sample points or the range don't have to be the same as for the other arguments.
All values are multiplied with *Fac2SI*. Default is *Fac2SI*=1.0.

.Arg1 =** ***min ... max
.Arg1.Fac2SI =** ***Fac2SI

Values of the second kinematics generalised coordinate. There should be at least five sample points.The number of sample points or the range don't have to be the same as for the other arguments.
All values are multiplied with *Fac2SI*. Default is *Fac2SI*=1.0.
Arg_1 has to be specified for every model that considers at least two generalised coordinates.

.Arg2 =** ***min ... max
.Arg2.Fac2SI =** ***Fac2SI

Values of the third kinematics generalised coordinate. There should be at least five sample points. The number of sample points or the range don't have to be the same as for the other arguments.
All values are multiplied with *Fac2SI*. Default is *Fac2SI*=1.0.
Arg_2 has to be specified for every model that considers three generalised coordinates.

In case of 2D+2D Arg_2 is used as first argument of the second 2D mapping. In this case Arg_3 is required as second argument of the second 2D mapping and would correspond to the third kinematics generalised coordinate. In the 2D+2D model Arg_0 and Arg_2 would describe the wheel compression generalised coordinate, but would have the flexibility of beeing scaled independently so as to change the objective effects of the wheel compression with respect to the opposite wheel compression and/or steering rack displacement.

.Arg3 =** ***min ... max
.Arg3.Fac2SI =** ***Fac2SI

Values of the third kinematics generalised coordinate. There should be at least five sample points. The number of sample points or the range don't have to be the same as for the other arguments.
All values are multiplied with *Fac2SI*. Default is *Fac2SI*=1.0.
Arg_3 has to be specified only in case of 2D+2D.

.Data.Name =** ***DataNameList

List of suspension coordinates.

The quantities are selected by the blank separated list *DataNameList* of the keys **tx**, **ty**, **tz**, **rx**, **ry**, **rz**, **ISpring**, **IDamp**, **IBuf**, **IStabi**. Unknown keys are skipped and the values from the corresponding column in the data table are ignored. Typically unknown or commented out keys should start with a "%" character.

***.Data.Offset =**	***OffsetList***
***.Data.dArg0 =**	***dData_dArg0_List***
***.Data.dArg1 =**	***dData_dArg1_List***
***.Data.Fac2SI =**	***Fac2SI_List***

Each data coordinate (t_x , t_y , ...) can have an offset and one or two linear gradients depending on whether you parametrise a 1D or 2D mapping. These values $dArg_0$ and $dArg_1$ are optional tuning parameters. They allow to put an offset on the gradient of the mappings. Figure 12.10: shows an example from Model Check where the original l_{spring} (dotted lines) is compared with an alternate l_{spring} where $dArg_0 = 0.2$ is set, which results in the solid lines. For t_y $dArg_0 = 0.05$ is set. Use Cases: Parametric Target Setting, Motion Ratio Change.

All values are multiplied with their factor *Fac2SI* from *Fac2SI_List.* The calculation of a suspension coordinate is done by the formula:

$$value = Data \cdot Fac2SI + Offset + \frac{dData}{dArg0} \cdot Arg0 \quad \text{or} \qquad \textbf{(EQ 73)}$$

$$value = Data \cdot Fac2SI + Offset + \frac{dData}{dArg0} \cdot Arg0 + \frac{dData}{dArg1} \cdot Arg1 \qquad \textbf{(EQ 74)}$$

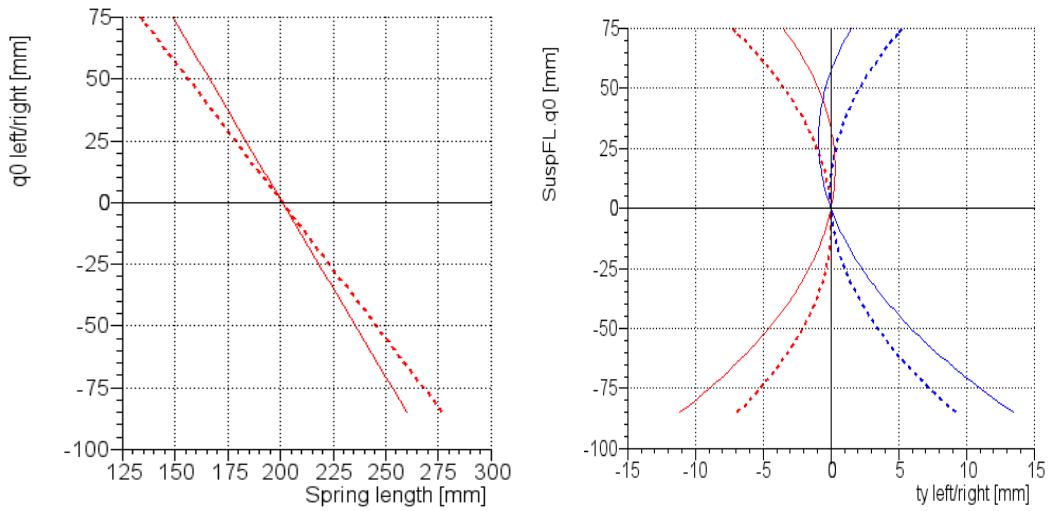


Figure 12.10: l_{spring} and t_y with gradient offset (solid) or without gradient offset (dottet)

***.Data:**	***TableWithValues***

Values for the selected coordinates from *DataNameList*. Each line contains the number of values, which are defined in *DataNameList*. The first DOF is kept constant, the second is varied first. Example for a 2D mapping (comp steer): The compression is kept constant while steering is varied. Then the same for the next compression value and so on.

***.Data1.Name =**	***DataNameList***
***.Data1.Offset =**	***OffsetList***
***.Data1.dArg1 =**	***dData_dArg1_List***
***.Data1.dArg2 =**	***dData_dArg2_List***
***.Data1.Fac2SI =**	***Fac2SI_List***
***.Data1:**	***TableWithValues***

Data to describe the second mapping to be superimposed on the first mapping.

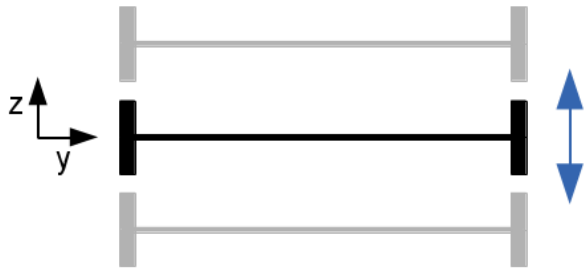
***.Data2.Name =**	***DataNameList***
***.Data2.Offset =**	***OffsetList***
***.Data2.dArg2 =**	***dData_dArg2_List***
***.Data2.dArg3 =**	***dData_dArg3_List***
***.Data2.Fac2SI =**	***Fac2SI_List***
***.Data2:**	***TableWithValues***

Data to describe the third mapping to be superimposed on the first two mappings.

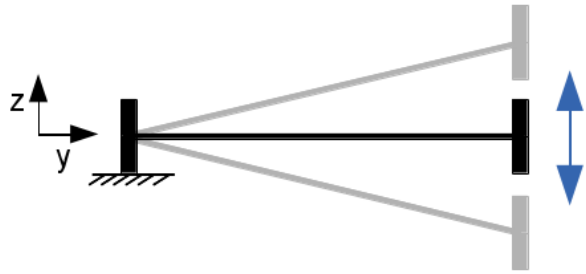
Measurement procedure

Below there is an outline of the measurement procedures, which are neccessary to parametrise SKC-files for the different models. The kinematic values (translations, rotations) are always measured at the left wheel. See also 12.1.2 for more information.

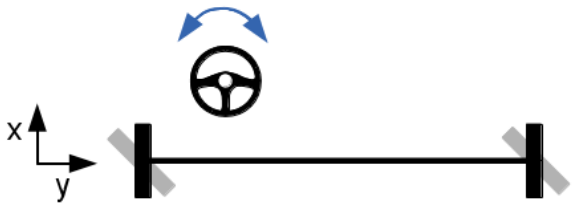
- **A**: Symmetrical compression (assumption: the compression of the right wheel has no influence) **(1D)**



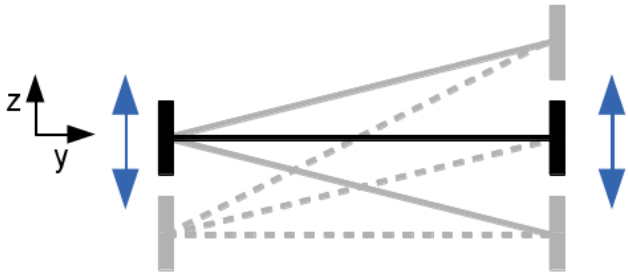
- **B**: Compression of the right wheel while compression of the left wheel remains zero **(1D)**



- **C**: Steering while compression of both wheels remains zero **(1D)**



- **D**: All combinations of compression of the left wheel and compression of the right wheel **(2D)**



- **E**: All combinations of symmetrical compression and steering **(2D)**

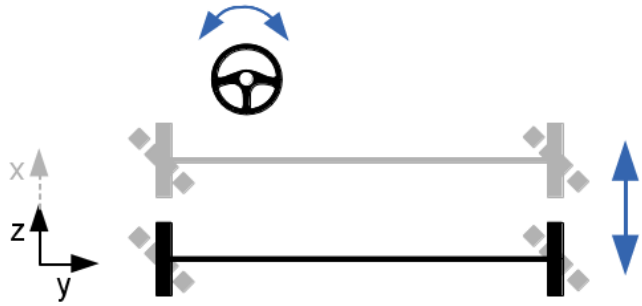


Table 12.3: Measurement procedures for all models

Model	Keyword combination	A	B	C	D	E
1D	comp	x				
1D + 1D	comp comp comp steer	x x	x	 x		
1D + 1D + 1D	comp comp steer	x	x	x		
2D	compsteer compcomp				 x	x
2D + 1D	compcomp steer compsteer comp		 x	x	x	 x
2D + 2D	compsteer compcomp				x	x

Concept of superposition

The following equations show how CarMaker calculates the kinematic values for the models, which use superposition. The values of the first mapping are always considered absolute. The values of a second or third mapping are calculated relative to the values of the first mapping. If there is an offset at zero in the mapping, CarMaker subtracts it once for each of the additional mappings. In the following equations k_0, k_1 and k_2 describe the nonlinear mappings.

- 1D+1D (comp comp)

$$\begin{bmatrix}\mathbf{t}\\ \mathbf{r}\end{bmatrix}_{kin} = \mathbf{k}_0(q_0) + \mathbf{k}_1(q_1 - q_0) - \mathbf{k}_1(q_0 = 0)$$

- 1D+1D (comp steer)

$$\begin{bmatrix}\mathbf{t}\\ \mathbf{r}\end{bmatrix}_{kin} = \mathbf{k}_0(q_0) + \mathbf{k}_1(q_2) - \mathbf{k}_1(q_2 = 0)$$

- 1D+1D+1D (comp comp steer)

$$\begin{bmatrix}\mathbf{t}\\ \mathbf{r}\end{bmatrix}_{kin} = \mathbf{k}_0(q_0) + \mathbf{k}_1(q_1 - q_0) + \mathbf{k}_2(q_2) - \mathbf{k}_1(q_1 = 0) - \mathbf{k}_2(q_2 = 0)$$

- 2D+1D (compcomp steer)

$$\begin{bmatrix}\mathbf{t}\\ \mathbf{r}\end{bmatrix}_{kin} = \mathbf{k}_0(q_0, q_1) + \mathbf{k}_1(q_2) - \mathbf{k}_1(q_2 = 0)$$

- 2D+1D (compsteer comp)

$$\begin{bmatrix}\mathbf{t}\\ \mathbf{r}\end{bmatrix}_{kin} = \mathbf{k}_0(q_0, q_2) + \mathbf{k}_1(q_1 - q_0) - \mathbf{k}_1(q_1 = 0)$$

- 2D+2D (compsteer compcomp)

$$\begin{bmatrix}\mathbf{t}\\ \mathbf{r}\end{bmatrix}_{kin} = \mathbf{k}_0(q_0, q_2) + \mathbf{k}_1(q_0, q_1) - \mathbf{k}_1(q_0, q_1 = q_0)$$

Example This parameter set is used for a steered front axle. The file contains the kinematic values of two 1D mappings depending on the DOFs wheel compression and steering rack displacement.

```
#INFOFILE1 - Do not remove this line!
FileIdent =                 CarMaker-SuspKnC-* 2
SuspF.Kin.N =               1
SuspF.Kin.0.Kind =          MapNL
SuspF.Kin.0.ValidSide =     left+right
SuspF.Kin.0.InputSide =     left

SuspF.Kin.0.L.Arg =         comp    steer
SuspF.Kin.0.L.Arg0 =   -0.1000   -0.0875   -0.0750   -0.0625   -0.0500   -0.0375   -0.0250
-0.0125   0.0000    0.0125    0.0250    0.0375    0.0500    0.0625    0.0750    0.0875    0.1000
SuspF.Kin.0.L.Arg0.Fac2SI = 1.0

SuspF.Kin.0.L.Arg1 =   -0.0800   -0.0700   -0.0600   -0.0500   -0.0400   -0.0300   -0.0200
-0.0100   0.0000    0.0100    0.0200    0.0300    0.0400    0.0500    0.0600    0.0700    0.0800
SuspF.Kin.0.L.Arg1.Fac2SI = 1.0

SuspF.Kin.0.L.Data.Name =   %i0   tx    ty    tz    rx    ry    rz    lSpring  lDamp   lStabi
SuspF.Kin.0.L.Data.Fac2SI =  1    1.0   1.0   1.0   1.0   1.0   1.0    1.0      1.0      1.0

SuspF.Kin.0.L.Data:
0  -0.004469 -0.009499 -0.093134 -0.012512 0.021530 0.001601 0.261752 0.582366 -0.090148
1  -0.003796 -0.007209 -0.081449 -0.008960 0.019060 0.001152 0.250772 0.571352 -0.078831
2  -0.003151 -0.005243 -0.069762 -0.005816 0.016537 0.000718 0.239732 0.560274 -0.067522
3  -0.002539 -0.003596 -0.058090 -0.003098 0.013961 0.000310 0.228643 0.549144 -0.056236
4  -0.001960 -0.002264 -0.046435 -0.000820 0.011325 -0.000059 0.217501 0.537956 -0.044973
5  -0.001416 -0.001242 -0.034797 0.001000 0.008620 -0.000374 0.206303 0.526707 -0.033727
6  -0.000907 -0.000526 -0.023178 0.002342 0.005837 -0.000623 0.195045 0.515391 -0.022490
7  -0.000435 -0.000113 -0.011578 0.003181 0.002967 -0.000794 0.183724 0.504005 -0.011252
8   0.000000 -0.000000  0.000000 0.003491 -0.000000 -0.000873 0.172337 0.492544 0.000000
9   0.000395 -0.000185  0.011556 0.003239 -0.003075 -0.000847 0.160881 0.481003 0.011280
10  0.000750 -0.000667  0.023089 0.002391 -0.006270 -0.000705 0.149352 0.469376 0.022608
11  0.001063 -0.001446  0.034597 0.000905 -0.009596 -0.000432 0.137749 0.457656 0.034004
12  0.001332 -0.002522 0.046082 -0.001266 -0.013068 -0.000014 0.126068 0.445836 0.045492
13  0.001554 -0.003895  0.057542 -0.004176 -0.016700 0.000563 0.114307 0.433905 0.057101
14  0.001729 -0.005570  0.068983 -0.007890 -0.020509 0.001314 0.102464 0.421850 0.068871
15  0.001853 -0.007551  0.080413 -0.012487 -0.024519 0.002258 0.090532 0.409648 0.080848
16  0.001925 -0.009849  0.091848 -0.018067 -0.028757 0.003416 0.078504 0.397260 0.093103

SuspF.Kin.0.L.Data.Name =   %i0   tx    ty    tz    rx    ry    rz    lSpring  lDamp   lStabi
SuspF.Kin.0.L.Data.Fac2SI =  1    1.0   1.0   1.0   1.0   1.0   1.0    1.0      1.0      1.0

SuspF.Kin.0.L.Data1:
0  -0.006768 -0.082342 -0.006540 -0.153790 -0.138477 0.781259 0.158501 0.478605 0.038193
1  -0.010649 -0.067216 -0.004865 -0.117734 -0.123141 0.639889 0.159995 0.480110 0.028317
2  -0.012138 -0.054722 -0.003649 -0.090320 -0.107020 0.525476 0.161602 0.481731 0.021111
3  -0.012199 -0.043698 -0.002693 -0.067886 -0.090330 0.424658 0.163280 0.483422 0.015441
4  -0.011226 -0.033674 -0.001917 -0.048931 -0.073146 0.331988 0.165012 0.485167 0.010853
5  -0.009424 -0.024407 -0.001279 -0.032683 -0.055501 0.244592 0.166788 0.486956 0.007118
6  -0.006909 -0.015758 -0.000756 -0.018698 -0.037420 0.160720 0.168603 0.488784 0.004109
7  -0.003754 -0.007639 -0.000332 -0.006697 -0.018913 0.079185 0.170454 0.490648 0.001750
8   0.000000 -0.000000 0.000000  0.003491 -0.000000 -0.000873 0.172337 0.492544 0.000000
9   0.004320  0.007193 0.000246  0.011976  0.019328 -0.080137 0.174254 0.494474 -0.001163
10  0.009193 0.013956  0.000409  0.018815 0.039056 -0.159163  0.176200 0.496434 -0.001740
11  0.014612 0.020291  0.000506  0.024028 0.059178 -0.238438  0.178162 0.498408 -0.001709
12  0.020585 0.026190  0.000508  0.027604 0.079674 -0.318409  0.180164 0.500423 -0.001076
13  0.027116 0.031633  0.000439  0.029505 0.100536 -0.399514  0.182178 0.502450  0.000214
14  0.034220 0.036591  0.000296  0.029663 0.121744 -0.482196  0.184204 0.504489  0.002201
15  0.041916 0.041021  0.000086  0.027979 0.143277 -0.566931  0.186227 0.506524  0.004944
16  0.050224 0.044867 -0.000168  0.024314 0.165114 -0.654257  0.188215 0.508523  0.008533
```

12.4 Compliance Models

All model parameters have a prefix, depending on the corresponding suspension and the number of the model definition. The prefix can be an empty string or a string, ending with a dot ("SuspF.", "SuspF.Kin.3." for kinematics, "SuspR.Com.2." for compliance).

ModelName	Description	DOF 1	DOF 2
Linear	forces/torques described by linear mappings	Frc./Trq. and/or FrcOpp/TrqOpp	
Coeff1D	forces/torques described by linear mappings, for different values of the suspension compression	compression	Frc./Trq.
Displace1D	forces/torques described by nonlinear mappings	Frc/Trq and/or FrcOpp/TrqOpp	
Displace2D	forces/torques described by nonlinear mappings, for different values of the suspension compressions or the steering rack positions, or forces/torques acting on the current and opposite wheelcarrier as 2D mapping	compression steering Frc./Trq.	Frc/Trq Frc/Trq FrcOpp/TrqOpp

12.4.1 “Linear Frame Fr1” and “Linear Frame Fr2”

Description

The compliance displacement is based on linear mappings, defined by constant coefficients for the forces and torques attacking the current and the opposite wheel carrier. In (EQ 75) "W" stands for wrench, which is a substitution for the different forces and torques

In model “Linear Frame Fr1” the forces are decomposed in frame Fr1.
In model “Linear Frame Fr2” the forces are decomposed in frame Fr2.

$$\begin{bmatrix} \mathbf{t} \\ \mathbf{r} \end{bmatrix} = \mathbf{c}_W \cdot \mathbf{W} + \mathbf{c}_{W_{opp}} \cdot \mathbf{W}_{opp} \qquad \text{(EQ 75)}$$

Parameters

The following parameters are required for this model:

ModelName	ModelKind	Description
Linear Frame Fr1	CoeffConstFr1	Compliance displacement due forces in frame Fr1
Linear Frame Fr2	CoeffConstFr2	Compliance displacement due forces in frame Fr2

The model can be parametrised in the VehicleDataSet GUI or in a SKC-File.

.Data.Name =** ***DataNameList

Names of the quantities being modified by compliance effects. The quantities are selected by the blank separated list *DataNameList* of the keys **tx**, **ty**, **tz**, **rx**, **ry**, **rz**. Unit: for tx, ty, tz [m/N] and [1/N]; for rx, ry, rz [rad/N] and [rad/Nm].

Remark: Usually it is meaningful to leave out forces in z direction. The displacements caused by Forces in z direction are considered in the kinematics due to q_0 displacement.

***.Frc.x =**	***Values***
***.Frc.y =**	***Values***
***.Frc.z =**	***Values***
***.FrcOpp.x =**	***Values***
***.FrcOpp.y =**	***Values***
***.FrcOpp.z =**	***Values***

Compliance coefficients depending on forces (along x, y, z axis) to the current (*.Frc.*) or the opposite (*.FrcOpp.*) wheel carrier. Point of attack is the wheel center. Unit: [N].

***.Frc.Fac2SI =**	***Fac2SI_List***
***.Trq.Fac2SI =**	***Fac2SI_List***

All coefficients are multiplied by the factors from *Fac2SI_List*, from *.Frc.Fac2SI for forces and from *.Trq.Fac2SI for torques.

***.Trq.x =**	***Values***
***.Trq.y =**	***Values***
***.Trq.z =**	***Values***
***.TrqOpp.x =**	***Values***
***.TrqOpp.y =**	***Values***
***.TrqOpp.z =**	***Values***

Compliance coefficients depending on torques (arround x, y, z axis) to the current (*.Trq.*) or the opposite (*.TrqOpp.*) wheel carrier. The torques each are reduced to the wheel center. Unit: [Nm].

***.FrcDamp =**	***Values***
***.FrcOppDamp =**	***Values***

Compliance coefficients depending on the damper force to the current (*.FrcDamp.*) or the opposite (*.FrcOppDamp.*) wheel carrier. Unit: [N].

12.4.2 “Coeff1DFr1” and “Coeff1DFr2”

Description

The compliance displacement is based on a linear, compression depending coefficients for the forces, attacking the current suspension.

In model “Coeff1DFr1” the forces are decomposed in frame Fr1.
In model “Coeff1DFr2” the forces are decomposed in frame Fr2.

Parameters

The following parameters are required for this model: :

ModelName	ModelKind	Description
Linear Frame Fr1	Coeff1DFr1	Compliance displacement due to forces in frame Fr1
Linear Frame Fr2	Coeff1DFr2	Compliance displacement due to forces in frame Fr2

The model can only be parametrised in a SKC-File.

.Arg0 =** ***min ... max
.Arg0.Fac2SI =** ***Fac2SI

Values of first degree of freedom (DOF), suspension compression.
All values are multiplied with *Fac2SI*. Default is *Fac2SI* = 1.0.

.Data.Name =** ***DataNameList

Names of the quantities being modified by compliance effects. The quantities are selected by the blank separated list *DataNameList* of the keys **tx**, **ty**, **tz**, **rx**, **ry**, **rz**, **ISpring**, **IDamp**, **IBuf**, **IStabi**. Unknown keys are skipped and the values from the corresponding column in the data table are ignored. Unknown or commented out keys typically should start with a "%" character.

.Frc.Fac2SI =** ***Fac2SI_List
.Trq.Fac2SI =** ***Fac2SI_List

All coefficients are multiplied by the factors from *Fac2SI_List*, from *.Frc.Fac2SI for forces and from *.Trq.Fac2SI for torques.

.Frc.x.Data :** ***TableWithValues
.Frc.y.Data :** ***TableWithValues
.Frc.z.Data :** ***TableWithValues
.Trq.x.Data :** ***TableWithValues
.Trq.y.Data :** ***TableWithValues
.Trq.z.Data :** ***TableWithValues

Compliance coefficients for the selected coordinates from *DataNameList* depending on forces (along x, y, z axis) or torques (arround x, y, z axis) to the current wheel carrier. Point of attack is the wheel center.

The table starts with values for the first element of *Arg0* end ends with values for the last one.

Unit: for tx, ty, tz [m/N] and [1/N]; for rx, ry, rz [rad/N] and [rad/Nm].

*.FrcDamp.Data : *TableWithValues*

Compliance coefficients depending on damper force to the current (*.Frc.*) wheel carrier.

Example This parameter set is for the rear axle and for the left suspension. For the right suspension the parameter set is mirrored. The compliance depends on Frc.x, Frc.y given in unit "N" and Trq.x, Trq.y, Trq.z given in unit “Nm“. It defines compliance coefficients for suspension coordinates tx, ty, rx, ry, rz, depending on wheel compression.

```
SuspR.Com.N =                 1
SuspR.Com.0.Kind =            CoeffConstFr1
SuspR.Com.0.ValidSide =       left+right
SuspR.Com.0.InputSide =       left

SuspR.Com.0.L.Arg0 =          -.100E+00 0.000E+00 0.100E+00
SuspR.Com.0.L.Arg0.Fac2SI =   1.0

SuspR.Com.0.L.Data.Name =     tx    ty rx    ry    rz
SuspR.Com.0.L.Frc.Fac2SI =    1.0   1.0 1.0   1.0   1.0
SuspR.Com.0.L.Trq.Fac2SI =    1.0   1.0 1.0   1.0   1.0

SuspR.Com.0.L.Frc.x.Data:
       0.170E-07 -0.165E-08-0.501E-08  0.798E-10 -0.146E-07
       0.160E-07 -0.164E-08-0.503E-08  0.796E-10 -0.148E-07
       0.180E-07 -0.166E-08-0.505E-08  0.794E-10 -0.150E-07

SuspR.Com.0.L.Frc.y.Data:
       -0.170E-08  0.561E-07 0.790E-07  0.495E-14  0.183E-08
       -0.164E-08  0.562E-07 0.794E-07  0.490E-14  0.181E-08
       -0.158E-08  0.563E-07 0.798E-07  0.485E-14  0.179E-08

SuspR.Com.0.L.Trq.x.Data:
       -0.501E-08  0.790E-07 0.410E-06 -0.130E-14  0.550E-08
       -0.503E-08  0.794E-07 0.411E-06 -0.123E-14  0.558E-08
       -0.505E-08  0.798E-07 0.412E-06 -0.116E-14  0.566E-08

SuspR.Com.0.L.Trq.y.Data:
       0.790E-10 -0.309E-14-0.470E-14  0.190E-05 -0.541E-14
       0.796E-10 -0.209E-14-0.461E-14  0.187E-05 -0.521E-14
       0.802E-10 -0.109E-14-0.452E-14  0.184E-05 -0.501E-14

SuspR.Com.0.L.Trq.z.Data:
       -0.158E-07  0.181E-080.559E-08  0.803E-18  0.190E-06
       -0.148E-07  0.181E-080.558E-08  0.703E-18  0.191E-06
       -0.138E-07  0.181E-080.557E-08  0.603E-18  0.192E-06
```

12.4.3 “Displace1DFr1” and “Displace1DFr2”

Description

The compliance displacement is described by nonlinear displacement mappings. The displacement mappings depend on the forces/torques attacking the current or the opposite wheel carrier.

In model “Displace1DFr1” the forces are decomposed in frame Fr1.
In model “Displace1DFr2” the forces are decomposed in frame Fr2.

Parameters

The following parameters are required for this model:

ModelName	ModelKind	Description
Displace 1D Fr1	Displace1DFr1	Compliance displacement due forces in frame Fr1
Displace 1D Fr2	Displace1DFr2	Compliance displacement due forces in frame Fr2

The model can only be parametrised in a SKC-File.

.Arg = *ArgName

The quantity, the displacement relationship depends on, is selected by *ArgName*. Known keys are **Frc.x**, **Frc.y**, **Frc.z**, **Trq.x**, **Trq.y**, **Trq,z**, **FrcDamp** for the current wheel carrier and **FrcOpp.x**, **FrcOpp.y**, **FrcOpp.z**, **TrqOpp.x**, **TrqOpp.y**, **TrqOpp,z**, **FrcDampOpp** for the opposite wheel carrier.

The forces (along x, y, z axis) or torques (arround x, y, z axis) attacks the current or opposite wheel carrier in the wheel center and effect on the displacement of the current wheel center.

Remark: Usually it is meaningful to leave out forces in z direction.

.Arg0 =** ***min ... max
.Arg0.Fac2SI =** ***Fac2SI

Values of the function argument.
All values are multiplied with *Fac2SI*. Default is *Fac2SI*=1.0.

.Data.Name =** ***DataNameList

The quantities are selected by the blank separated list *DataNameList* of the keys **tx**, **ty**, **tz**, **rx**, **ry**, **rz**, **ISpring**, **IDamp**, **IBuf**, **IStabi**. Unknown keys are skipped and the values from the corresponding column in the data table are ignored. Typically unknown or commented out keys should start with a "%" character.

.Data :** ***TableWithValues

Displacement values for the coordinates from *DataNameList.*

The table starts with values for the first element of *Arg0* end ends with values for the last one.

Unit: for tx, ty, tz, lSpring, lDamp, lBuf, lStabi [m]; for rx, ry, rz [rad].

*.Data.Fac2SI = *Fac2SI_List*

All data columns are multiplied by the factors from *Fac2SI_List*.

Example This parameter set is for the front axle and for the left suspension only. The compliance depends on Frc.x, given in unit "kN". It defines compliance effects for suspension coordinates ty and rz, both given in SI units.

```
SuspF.Com.0.Kind =            Displace1DFr1 1
SuspF.Com.0.ValidSide =       left+right
SuspF.Com.0.InputSide =       left
SuspF.Com.0.L.Arg =           Frc.x
SuspF.Com.0.L.Arg0 =          0.0  1.0 2.0 3.0 4.0 4.5 4.6 4.7
SuspF.Com.0.L.Arg0.Fac2SI =   1.0e3
SuspF.Com.0.L.Data.Name =     ty     rz
SuspF.Com.0.L.Data.Fac2SI =   1.0    1.32456
SuspF.Com.0.L.Data:
   0.000      0.000
   0.001      0.006
   0.002      0.011
   0.004      0.015
   0.008      0.018
   0.016      0.020
   0.032      0.021
   0.032      0.021
```

12.4.4 “Displace2DFr1” and “Displace2DFr2”

Description

The compliance displacement is described by nonlinear displacement mappings. The displacement mappings depend on the wheel compression or the steering rack position on the current wheel carrier and/or forces/torques, attacking the current and/or the opposite wheel carrier.

In model “Displace2DFr1” the forces are decomposed in frame Fr1.
In model “Displace2DFr2” the forces are decomposed in frame Fr2.

Parameters

The following parameters are required for this model:

ModelName	ModelKind	Description
Displace 1D Fr1	Displace2DFr1	Compliance displacement due forces in frame Fr1
Displace 1D Fr2	Displace2DFr2	Compliance displacement due forces in frame Fr2

The model can only be parametrised in a SKC-File.

.Arg = *NameDoF0* *NameDoF1

The first name *NameDoF0* can be “**comp**” for the wheel compression or “**steer**” for steering rack position or **Frc.x**, **Frc.y**, **Frc.z**, **Trq.x**, **Trq.y**, **Trq.z**, **FrcDamp** as the first degree of freedom. The second name *NameDoF1* defines the second degree of freedom, the displacement relationship depends on. Known keys are **Frc.x**, **Frc.y**, **Frc.z**, **Trq.x**, **Trq.y**, **Trq.z**, **FrcDamp** for the current suspension and **FrcOpp.x**, **FrcOpp.y**, **FrcOpp.z**, **TrqOpp.x**, **TrqOpp.y**, **TrqOpp.z**, **FrcDampOpp** for the opposite suspension.

This means, there are five possible keyword-combinations as Input for **.Arg*:

```
  comp     <wrench>
  comp     <wrenchOpp>
  steer    <wrench>
  steer    <wrenchOpp>
<wrench>  <wrenchOpp>
```

"wrench" and "wrenchOpp" are substitutions for the different forces/torques.

The forces (along x, y, z axis) or torques (arround x, y, z axis) attack the current or opposite wheel carrier in the wheel center and have an effect on the displacement of the current wheel center.

.Arg0 = *min ... max
.Arg0.Fac2SI = *Fac2SI

Sample points for wheel compression, steering rack position or wrench.
All values are multiplied with *Fac2SI*. Default is *Fac2SI*=1.0.

.Arg1 = *min ... max*
.Arg1.Fac2SI = *Fac2SI

Sample points for force or torque.
All values are multiplied with *Fac2SI*. Default is *Fac2SI*=1.0.

*.Data.Name = *DataNameList*

The quantities are selected by the blank separated list *DataNameList* of the keys **tx**, **ty**, **tz**, **rx**, **ry**, **rz**, **ISpring**, **IDamp**, **IBuf**, **IStabi**. Unknown keys are skipped and the values from the corresponding column in the data table are ignored. Typically unknown or commented out keys should start with a "%" character.

*.Data : *TableWithValues*

Displacement values for the selected coordinates from *DataNameList*. Ordering: Each line contains the number of values, which are defined in *DataNameList*. The first DOF is kept constant, the second is varied first. Example: The compression is kept constant while Frc.x is varied. Then the same for the next compression and so on.
Unit: for tx, ty, tz, ISpring, IDamp, IBuf, IStabi [m]; for rx, ry, rz [rad].

*.Data.Fac2SI = *Fac2SI_List*

All data columns are multiplied by the factors from *Fac2SI_List*.

Example This parameter set is for the rear axle and for the left suspension. For the right suspension the parameter set is mirrored. The compliance depends on Frc.x and Frc.y, given in unit "N" and the wheel compression, given in unit “m”. It defines the compliance displacements for suspension coordinates tx, ty, tz, rx, ry, rz.

```
SuspR.Com.N =                2

SuspR.Com.0.Kind =            Displace2DFr1 1
SuspR.Com.0.ValidSide =       left+right
SuspR.Com.0.InputSide =       left

SuspR.Com.0.L.Arg =           comp Frc.x

SuspR.Com.0.L.Arg0 =         -.100E+00 -.050E+00 0.000E+00 0.050E+00 0.100E+00
SuspR.Com.0.L.Arg0.Fac2SI =   1.0
SuspR.Com.0.L.Arg1 =         -.850E+05 -.425E+05 0.000E+00 0.425E+05 0.850E+05
SuspR.Com.0.L.Arg1.Fac2SI =   1.0

SuspR.Com.0.L.Data.Name =     %i0    %i1    tx    ty    tz    rx    ry    rz
SuspR.Com.0.L.Data.Fac2SI =   1      1      1.0   1.0   1.0   1.0   1.0   1.0

SuspR.Com.0.L.Data:
   0 0    -0.155E-02  0.151E-03  0.710E-03  0.430E-03 -0.788E-05  0.146E-02
   0 1    -0.775E-03  0.075E-03  0.355E-03  0.215E-03 -0.003E-03  0.730E-03
   0 2    0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00
   0 3    0.780E-03 -0.084E-03 -0.392E-03 -0.233E-03  0.003E-03 -0.730E-03
   0 4    0.156E-02 -0.168E-03 -0.784E-03 -0.467E-03  0.597E-05 -0.146E-02
   ...
   4 0    -0.115E-02  0.111E-03  0.670E-03  0.390E-03 -0.748E-05  0.106E-02
   4 1    -0.575E-03  0.055E-03  0.335E-03  0.195E-03 -0.004E-03  0.530E-03
   4 2    0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00
   4 3    0.580E-03 -0.064E-03 -0.372E-03 -0.213E-03  0.003E-03 -0.530E-03
   4 4    0.116E-02 -0.128E-03 -0.744E-03 -0.427E-03  0.557E-05 -0.106E-02


SuspR.Com.1.Kind =            Displace2DFr1 1
```

```
SuspR.Com.1.ValidSide =       left+right
SuspR.Com.1.InputSide =       left

SuspR.Com.1.L.Arg =           comp Frc.y

SuspR.Com.1.L.Arg0 =         -.100E+00 -.050E+00 0.000E+00 0.050E+00 0.100E+00
SuspR.Com.1.L.Arg0.Fac2SI =   1.0
SuspR.Com.1.L.Arg1 =         -.850E+05 -.425E+05 0.000E+00 0.425E+05 0.850E+05
SuspR.Com.1.L.Arg1.Fac2SI =   1.0

SuspR.Com.1.L.Data.Name =     %i0    %i1    tx    ty    tz    rx    ry    rz
SuspR.Com.1.L.Data.Fac2SI =   1      1      1.0   1.0   1.0   1.0   1.0   1.0

SuspR.Com.1.L.Data:
   ...
```

12.4.5 “SetZero”

Description Model to suppress previous displacements for selected quantities

$$\begin{bmatrix} \mathbf{t} \\ \mathbf{r} \end{bmatrix} = 0 \qquad \text{(EQ 76)}$$

Parameters The following parameters are required for this model:

.Data.Names =** ***DataNamesList

The suspension displacements by compliance effects of the selected quantities are set to zero.

The quantities are selected by the blank separated list *DataNameList* of the keys **tx**, **ty**, **tz**, **rx**, **ry**, **rz**, **lSpring**, **lDamp**, **lBuf**, **lStabi**. Unknown keys are skipped and the values from the corresponding column in the data table are ignored. Typically unknown or commented out keys should start with a "%" character.

Example

```
SuspR.Com.0.Kind =            SetZero 1

SuspR.Com.0.L.Data.Names =    tx    rx
```

12.4.6 Example: Superposition of Compliance Models

```
SuspF.Com.N = 4
#--- "basic" compliance parameters by Model 0
SuspF.Com.0.Kind =     CoeffConstFr1
SuspF.Com.0.ValidSide =left+right        # Model 0 is used for left and right
SuspF.Com.0.InputSide =left              # parameters are given only for left
SuspF.Com.0.L.Data.Name =tx ty tz ry     # right by mirrowing left->right
SuspF.Com.0.....

#--- "extended", detailed parameters by Models 1, 2, 3

# Parametrization for right suspension based on parameters for
# a left suspension for model 1. Left parameters are "transfered" to right here.
#
SuspF.Com.1.Kind =     Displace1DFr1 1
SuspF.Com.1.ValidSide =left+right
SuspF.Com.1.InputSide =left

SuspF.Com.1.L.Arg =    comp Frc.x
SuspF.Com.1.L.....

#Parametrization for left and right suspension separately for model 2 and 3.

SuspF.Com.2.Kind =     Displace1DFr1 1
SuspF.Com.2.ValidSide =left+right
SuspF.Com.2.InputSide =left+right
SuspF.Com.2.L.Arg =    comp Frc.y
SuspF.Com.2.L.....
SuspF.Com.2.R.Arg =    comp Frc.y
SuspF.Com.2.R.....


SuspF.Com.3.Kind =     Displace1DFr1 1
SuspF.Com.3.ValidSide =left+right
SuspF.Com.3.InputSide =left+right
SuspF.Com.3.L.Arg =    comp Trq.z
SuspF.Com.3.L.....
SuspF.Com.3.R.Arg =    comp Trq.z
SuspF.Com.3.R.....
```

In this example the linear part is specified in model 0 ("basic" compliance). For all other models the specification should contain only the difference to model 0.

An external kinematics C code interface is now available, declared in the *include/Car/Susp.h* header file.

12.5 Suspension Interface

12.5.1 Overview

This chapter describes the global suspension interface from the headerfile `/include/Car/Susp.h`.

There are two interface structs defined for the information exchange between the simulation program and the suspension model, one for the initialisation and one for the evaluation function.

Interface for the initialisation function

Following table represents the input and output variables of the initialisation interface struct *tSuspCfgIF*:

Input Variable	Information
SuspNo	Identification number of the suspension side for vehicle, trailers
WheelBdy TwinWheelBdy WCBdy WheelBdyOpp TwinWheelBdyOpp WCBdyOpp	Body structs containing position in vehicle frame FrD, mass, inertia of wheel, twin wheel and wheel carrier bodies (current and opposite side)
TireInf TireInfOpp	Infofile handle of the tires (current and opposite side)

Output Variable	Information
DoubleSided	Specifies if the suspension model handles the left and right suspension side at once.
JointType	Specifies the rotation sequence of the joint between the wheel carrier and the vehicle. Currently supported are ZXY and YZX.
FrcFrame	Specifies in which frame the forces and torques act on the wheel carrier. Currently supported are vehicle frame Fr1, wheel carrier frame Fr2 and tire frame FrW.
Use_qSteer	Indicates if the suspension model needs the generalized coordinate for the steering rack travel as input.
Use_qCompOpp	Indicates if the suspension model needs the generalized coordinate for the opposite wheel compression (dependent axles) as input.

Interface for the evaluation function

Following table represents the input and output variables of the evaluation interface struct *tSuspIF*. The evaluation function always has two struct pointers *IFMain* and *IFOpp*. *IFMain* stands for the suspension side with the predefined SuspNo and *IFOpp* is the opposite side of SuspNo.

Input Variable	Unit	Information
Frc2WC[x,y,z]	N	Forces (tire, brake, driveline, external, gyro) applied to the wheel carrier center, expressed in specified frame
Trq2WC[x,y,z]	Nm	Torques (tire, brake, driveline, external, gyro) applied to the wheel carrier center, expressed in specified frame

Input Variable	Unit	Information
FrcSpring	N	Spring force along the spring thrust axis
FrcDamp	N	Damper force along the damper thrust axis
FrcStabi	N	Stabilizer bar force (stabilizer twist torque * lever arm)
FrcBuf	N	Buffer force
qComp qpComp qppComp	m m/s m/s^2	Generalized coordinate for wheel compression
qSteer qpSteer qppSteer	m m/s m/s^2	Generalized coordinate for steering rack travel

Output Variable	Unit	Information
Kin[tx,ty,tz,rx,ry,rz, lSpring,lDamp, lBuf, lStabi]	m,rad	Wheel motion vector (wheel translation $\vec{t}^{kin}$ in vehicle frame, wheel rotation $\vec{r}^{kin}$ with specified sequence and suspension lengths l^{kin}) due to kinematics
Com[tx...lStabi]	m,rad	Wheel motion vector due to compliance
dqComp[tx...lStabi] dqSteer[tx...lStabi] dqCompOpp[tx..lStabi]	m/m, rad/m	kinematic gradients / partial derivates of kinematic wheel motion vector $\partial\vec{t}^{kin}/\partial q_n$, $\partial\vec{r}^{kin}/\partial q_n$, $\partial l^{kin}/\partial q_n$ (with n=Comp, CompOpp and Steer)
FrcSecSpring	N	Additional generalized force due to secondary spring force

12.5.2 Kinematic characteristic parameters

With the information from the suspension interface, which contains among other the wheel carrier position and the kinematic gradients (partial derivation of wheel motion with respect to wheel compression or steering rack travel), the wheel instant axis can be determined. With the instant axis for steering (steering or kingpin axis) or the instant axis for wheel compression, several kinematic characteristic parameters can be derived.

Steering axis

The instant axis for steering rack travel or the steering axis is calculated using the partial derivation of wheel rotation with respect to steering rack travel $\partial\vec{r}_{ij}^{kin}/\partial q_{Steer}$ and the Hamel matrix *H*. Below, the calculation for the front left wheel is demonstrated:

$$\left[\vec{\Pi}_{2RFL,\,q_{Steer}}\right]_1 = \left[H\right]\left[\frac{\partial\vec{r}_{FL}^{kin}}{\partial q_{Steer}}\right]_1 = \begin{bmatrix}\cos(r_z) & -\sin(r_z)\cos(r_x) & 0\\ \sin(r_z) & \cos(r_z)\cos(r_x) & 0\\ 0 & \sin(r_x) & 1\end{bmatrix}\left[\frac{\partial\vec{r}_{FL}^{kin}}{\partial q_{Steer}}\right]_1 \qquad \textbf{(EQ 77)}$$

For the location of the axis a point *A* on the axis is searched. The vector from the wheel carrier center to the point *A* is calculated as follows:

$$\overrightarrow{WCA} = \frac{\left[\vec{\Pi}_{2RFL,\,q_{Steer}}\right]_1 \times \left[\frac{\partial\vec{t}_{FL}^{kin}}{\partial q_{Steer}}\right]_1}{\left[\vec{\Pi}_{2RFL,\,q_{Steer}}\right]_1^{\,2}} \qquad \textbf{(EQ 78)}$$

$\partial\vec{t}_{ij}^{kin}/\partial q_{Steer}$ is the partial derivation of wheel translation with respect to steering rack travel.

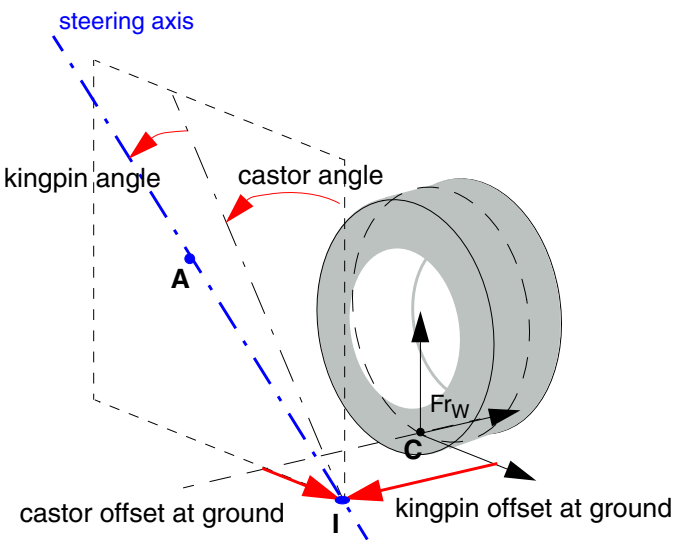


Figure 12.11: Steering axis

With the determined location and orientation of the steering axis, some important characteristic parameters can now be evaluated (see Figure 12.11).
The castor angle is the angle between the vehicle vertical axis and the projected steering axis on the vehicle x-z plane.The kingpin angle is the angle between the vehicle vertical axis and the projected steering axis on the vehicle y-z plane.
For the both offset parameters firstly the intersection point *I* of the steering axis with the road plane is calculated. The kingpin offset at ground is the projected vector $\overrightarrow{CI}$ on the y-axis of the tire frame Fr_W. The castor offset at ground is the projected vector $\overrightarrow{CI}$ on the x-axis of the tire frame Fr_W.

Roll and pitch center

In the same manner like for the steering axis, for each tire there is an instantaneous axis of rotation for the wheel jounce and rebound. Figure 12.12 illustrates an instantaneous axis of rotation of the rear left tire.

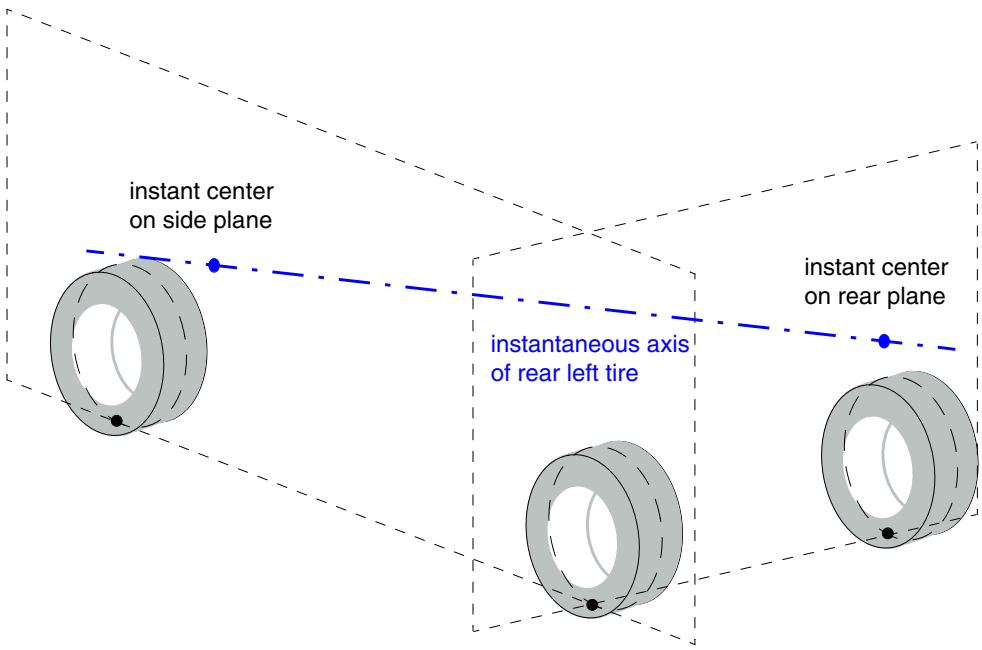


Figure 12.12: Instantaneous axis of rotation

The orientation and the location of the axis are calculated like for the steering axis case:

$$\left[\vec{\Pi}_{2RRL,\,q_{0RL}}\right]_1 = \left[H\right]\left[\frac{\partial \check{r}_{RL}^{kin}}{\partial q_{0RL}}\right] = \begin{bmatrix} \cos(r_z) & -\sin(r_z)\cos(r_x) & 0 \\ \sin(r_z) & \cos(r_z)\cos(r_x) & 0 \\ 0 & \sin(r_x) & 1 \end{bmatrix}\left[\frac{\partial \check{r}_{RL}^{kin}}{\partial q_{0RL}}\right] \qquad \textbf{(EQ 79)}$$

$$\overrightarrow{WCA} = \frac{\left[\vec{\Pi}_{2RRL,\,q_{0RL}}\right]_1 \times \left[\frac{\partial \check{t}_{RL}^{kin}}{\partial q_{0RL}}\right]_1}{\left[\vec{\Pi}_{2RRL,\,q_{0RL}}\right]_1^{\,2}} \qquad \textbf{(EQ 80)}$$

For each tire, two intersection points are calculated: an intersection point of the axis with the vertical side plane through the front and rear contact points and an intersection point of the axis with vertical front/rear plane through the left and right contact points.

Regarding the Figure 12.13, the roll center is found out by crossing the lines between the instant point on the front/rear plane and the corresponding contact point. The y-z position of the roll center is expressed in the plane frame. The frame origin is in the middle between the both contact points.

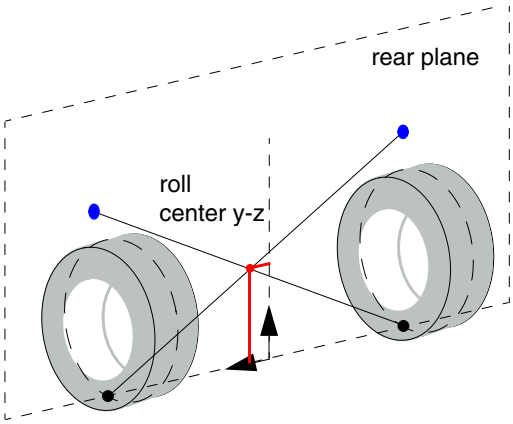


Figure 12.13: Calculation of roll center

In the same manner like the roll center, the pitch center is found out by crossing the lines between the instant point on the side plane and the corresponding contact point (see Figure 12.14). The x-z position of the pitch center is expressed in the plane frame. The frame origin is in the rear contact point.

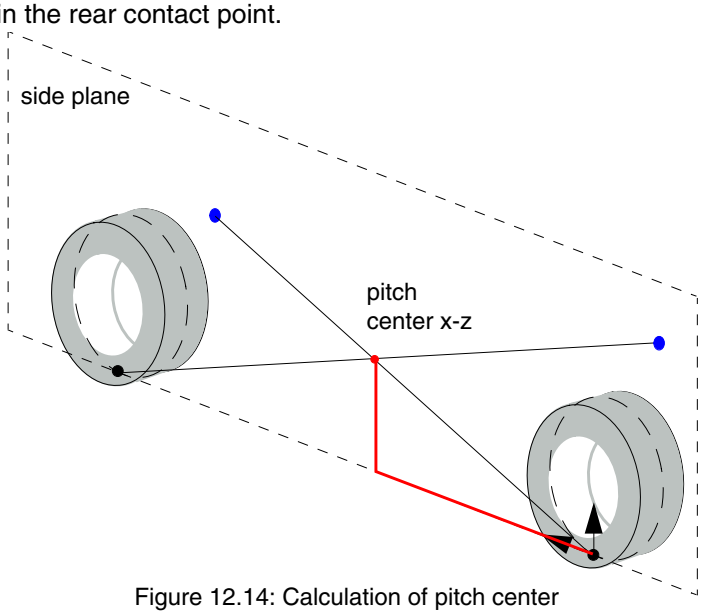


Figure 12.14: Calculation of pitch center

12.6 MBS Suspension

12.6.1 Overview

Besides the linear and non-linear kinematics and compliance mapping models described in the precedent chapter, the CarMaker suspension library offers some high-resoluted multi-body system (MBS) suspension models for the front and rear axle.

Each MBS suspension model is realized as a virtual suspension testbench with a fixed vehicle body. The suspension model runs besides the vehicle model and interacts with the vehicle model via the global suspension interface (see Figure 12.15). The interface for the MBS suspension model is the same as for the mapping based kinematics and compliance models (for details please refer to section 12.5 'Suspension Interface').

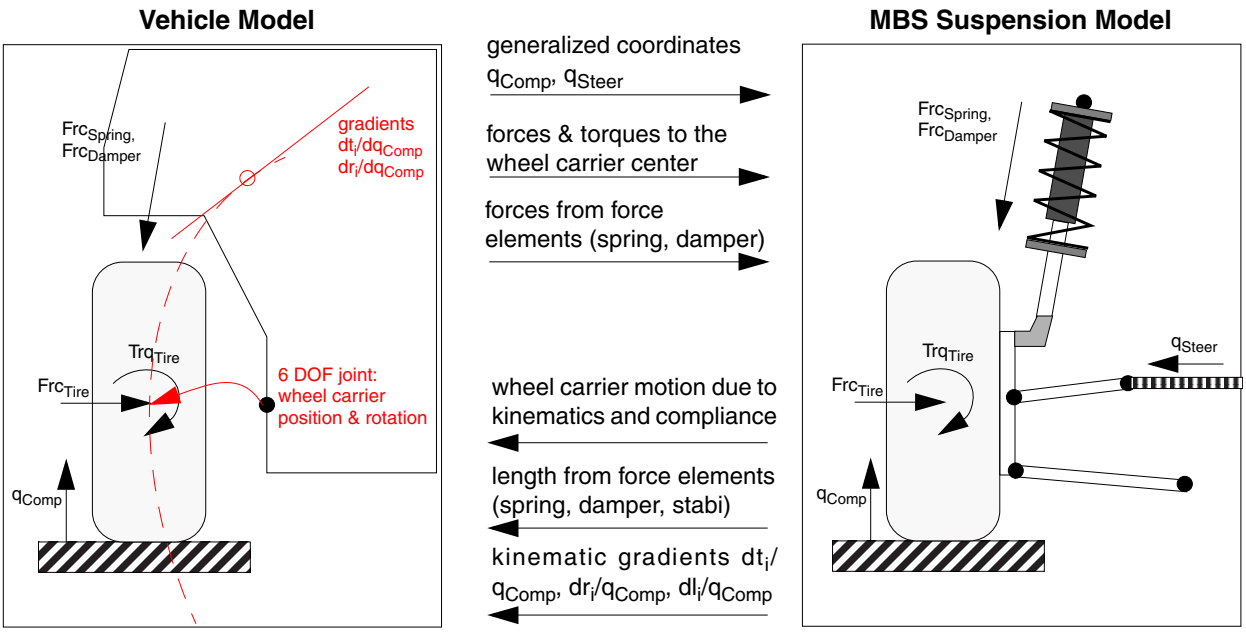


Figure 12.15: MBS suspension model interface with the vehicle model

The suspension model gets as input the generalized coordinates (wheel compression q_{Comp}, steering rack travel q_{Steer}), the acting forces and torques to the wheel carrier center (tire force and torque, brake and driveline moment, gyroscopic moment) and the forces from the suspension force elements (spring, damper, stabilizer, buffer).

As output, the suspension model delivers the same information as the mapping based kinematics and compliance models: wheel carrier motion (three translations $\check{t}_i^{kin}$, $\check{t}_i^{com}$ and three rotations $\check{r}_i^{kin}$, $\check{r}_i^{com}$) as kinematics and as compliance part, length of force elements $\check{l}_i^{kin}$ (spring, damper, stabilizer, buffer), kinematical gradients $\frac{\partial \check{t}_i^{kin}}{\partial q_{Comp}}$, $\frac{\partial \check{r}_i^{kin}}{\partial q_{Comp}}$, $\frac{\partial l_i^{kin}}{\partial q_{Comp}}$ for the wheel compression and depending on the model also for the opposite wheel compression and steering rack travel.

The MBS suspensions use the coordinate system $\mathtt{Fr}_{\mathtt{Susp}}$ (suspension frame). Its origin is located at the center of the axis which joins the two wheel centers of the left and right wheel for the axle in focus. For example, $\mathtt{Fr}_{\mathtt{Susp}}$ for the front axle would therefore be the center of the axis that joins the front left and front right wheel centers with respect to `FrD`.

Mapping generation

In the preparation phase, the MBS suspension model uses a similar kinematics and compliance procedure like the IPGKinematics tool to determine a non-linear two-dimensional kinematics and compliance mapping for the wheel carrier motion and the force elements length. This generated and in the background stored mapping is used for the following reasons:

- Modelcheck functionality
- Wheel travel in the static equilibrium phase shortly before the start of the simulation
- Determination of the kinematical gradients during the simulation
- Seperation of the total wheel motion into the kinematics and compliance part
- *Mapping* mode (description follows below)

Calculation mode

With the MBS suspension model, the suspension output (wheel carrier translation and rotation, kinematic gradients, length of force elements) can be determined during the simulation phase on two ways: *Dynamic* or *Mapping* calculation mode.

With the *Dynamic* mode, the wheel carrier motion is determined by the time integration of the model specific generalized coordinate accelerations $\ddot{q}_0 \ldots \ddot{q}_{N-1}$. The benefit of this mode is the considered suspension dynamics and that all input signals (wheel carrier forces and torques, wheel compression and steering rack travel) acts simultaneously like in the reality. The disadvantage is the high performance loss.

With the *Mapping* mode, the wheel carrier motion is determined directly from the stored mappings without time integration. The benefit of this mode is the fast computation time. The disadvantage is that the wheel carrier motion is like for the mapping based suspension models a superposition of several motion parts depending on only two independent input parameters (e.g.: wheel compression and side force + wheel compression and traction force). Using this mode the suspension dynamics (natural oscillations, damping effects) are not considered.

12.6.2 General Parameters

Following parameters are used for each MBS suspension model.

CalcMode = ***ModeString***

Specifies the calculation mode during the simulation phase. Possible modes are:

`Dyn`	Dynamic: wheel carrier motion determined via time integration (default)
`Map`	Mapping: wheel carrier motion from the stored mappings

RefPointInputSystem = ***x*** ***y*** ***z***

Specifies the origin of the vehicle frame `FrD` (in suspension own `FrSusp` coordinates). The coordinates XYZ point from the origin of `FrSusp` to the origin of `FrD`. Unit: [m].

CarrierL.StaticCamber = ***angle***
CarrierR.StaticCamber = ***angle***

Specifies the static camber angle for left and right side. Unit: [deg]. Default: 0.

CarrierL.StaticToe = ***angle***
CarrierR.StaticToe = ***angle***

Specifies the static toe angle for left and right side. Unit: [min]. Default: 0.

CarrierL.mass = ***mass***
CarrierR.mass = ***mass***
CarrierL.I = ***Ixx*** ***Iyy*** ***Izz***
CarrierR.I = ***Ixx*** ***Iyy*** ***Izz***

Specifies the left and right wheel carrier body mass [kg] and inertia (main diagonal elements only) [kgm^2] including the brake parts. The center of mass is calculated internally on the basis of bushing and joint positions.

SpringFrcL_Upp.pos = ***x*** ***y*** ***z***
SpringFrcR_Upp.pos = ***x*** ***y*** ***z***

Position of left and right upper coil spring point expressed in `FrSusp` coordinates. Unit: [m].

SpringFrcL_Low.pos = ***x*** ***y*** ***z***
SpringFrcR_Low.pos = ***x*** ***y*** ***z***

Position of left and right lower coil spring point expressed in `FrSusp` coordinates. Unit: [m].

DamperFrcL_Upp.pos = ***x*** ***y*** ***z***
DamperFrcR_Upp.pos = ***x*** ***y*** ***z***

Position of left and right upper damper point expressed in `FrSusp` coordinates. Unit: [m].

DamperFrcL_Low.pos = ***x*** ***y*** ***z***
DamperFrcR_Low.pos = ***x*** ***y*** ***z***

Position of left and right lower damper point expressed in `FrSusp` coordinates. Unit: [m].

Bushing parameters

Each following MBS suspension model uses except rigid joints also elastic bushing components, especially between the wishbones and the vehicle. Following parameters are used by each bushing component and will be explained here once:

<BushingPre>.c_x = ***ConstStiffness***
<BushingPre>.c_x: ***DataTable***

The bushing spring characteristic can be defined with a constant or with a 1D-lookup table describing the non-linear spring behavior. This characteristic translates compression along the bushing x-axis to spring force. Unit: [N/m].

Syntax
Infofile table mapping with 2 columns
<compression [m]> <bushing force [N]>

Example
Vehicle.Bush2DampL.c_x:

-0.0005	-1500
0	0
0.001	1500

Example
Vehicle.Bush2DampL.c_x = 1.5e6

<BushingPre>.c_y = ***ConstStiffness***
<BushingPre>.c_y: ***DataTable***
<BushingPre>.c_z = ***ConstStiffness***
<BushingPre>.c_z: ***DataTable***

The spring characteristic along the y- and z-axis are defined in same manner like for x-axis.

<BushingPre>.c_x.Amplify = *Factor*
<BushingPre>.c_y.Amplify = *Factor*
<BushingPre>.c_z.Amplify = *Factor*

Bushing spring amplification factor *Factor* along the x-, y- and z-axis. Only active if the spring characteristic is a 1D-lookup table. Unit [-]. Default: 1.0.

<BushingPre>.k = *x* *y* *z*

Specifies the three bushing damping characteristic values along the bushing x-, y- and z-axis. This characteristic translates compression velocity around the bushing axis to spring force. Unit: [Ns/m].

<BushingPre>.cT = *x* *y* *z*

Optional. 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. Unit: [Nm/rad]. Default: 0 0 0.

<BushingPre>.kT = *x* *y* *z*

Optional. 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. Unit: [Nms/rad]. Default: 0 0 0.

<BushingPre>.Trq_KO = *x* *y* *z*

Optional. Specifies the bushing preload torque around the bushing x-, y- and z-axis.
Unit: [Nm]. Default: 0 0 0.

<BushingPre>.pos = *x* *y* *z*

Specifies the position of the bushing center expressed in `FrSusp` coordinates (please refer to Figure 12.16). Unit: [m].

<BushingPre>.pntaxis = *x* *y* *z*

Specifies the position of a point locating on the bushing x-axis expressed in `FrSusp` coordinates (please refer to Figure 12.16). Unit: [m].

<BushingPre>.pntplane = ***x y z***

Specifies the position of a point locating on the bushing xy-plane expressed in `FrSusp` coordinates (please refer to Figure 12.16). Unit: [m].

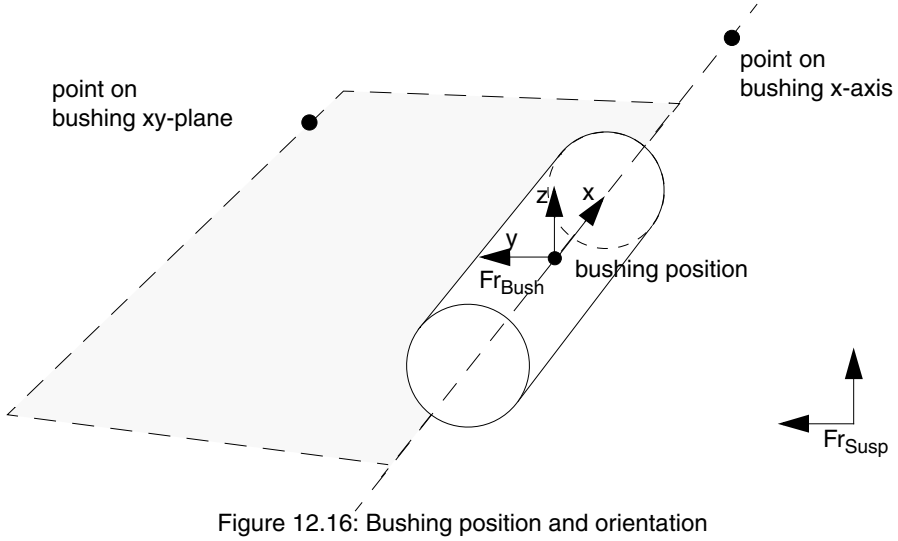


Figure 12.16: Bushing position and orientation

Mapping parameters

Cfg.qComp_max = ***maximum compression***

Specifies the maximum compression value of the kinematics mapping. The minimum compression has the same value with negative sign. The range of the kinematics mapping goes from minimum to maximum compression. Unit: [m]. Default: 0.1.

Cfg.nSteps_qComp = ***number of grid points***

Specifies the number of grid points of the kinematics mapping for compression. Unit: [-]. Default: 17.

Cfg.qSteer_max = ***maximum steering rack travel***

Specifies the maximum steering rack travel value of the kinematics mapping. The minimum steering rack travel has the same value with negative sign. The range of the kinematics mapping goes from minimum to maximum steering rack travel. Unit: [m]. Default: 0.08.

Cfg.nSteps_qSteer = ***number of grid points***

Specifies the number of grid points of the kinematics mapping for steering rack travel. Unit: [-]. Default: 17.

Cfg.qCompC_max = ***maximum compression***

Specifies the maximum compression value of the compliance mapping. The minimum compression has the same value with negative sign. The range of the compliance mapping goes from minimum to maximum compression. Unit: [m]. Default: 0.1.

Cfg.nSteps_qCompC = ***number of grid points***

Specifies the number of grid points of the compliance mapping for compression. Unit: [-]. Default: 7.

Cfg.Frc_max = ***maximum force***

Specifies the maximum force value of the compliance mapping. The minimum force has the same value with negative sign. The range of the compliance mapping goes from minimum to maximum force. Unit: [N]. Default: 7000.0.

Cfg.Trq_max = ***maximum torque***

Specifies the maximum torque value of the compliance mapping. The minimum torque has the same value with negative sign. The range of the compliance mapping goes from minimum to maximum torque. Unit: [Nm]. Default: 2000.0.

Cfg.nSteps_Frc = ***number of grid points***

Specifies the number of grid points of the compliance mapping for the force and the torque. Unit: [-]. Default: 11.

12.6.3 McPherson suspension

The McPherson wheel suspension is among the wheel-guiding spring and shock absorber strut axes. One speaks of a suspension strut when the spring forces are underpinned at the top by the piston rod mount and at the bottom by the damper external tube. The wheel-guiding suspension strut has the main advantage that all components involved in the suspension process and wheel guidance can be grouped into a single unit. Figure 12.17 is a schematic illustration of a McPherson suspension on the left vehicle side.

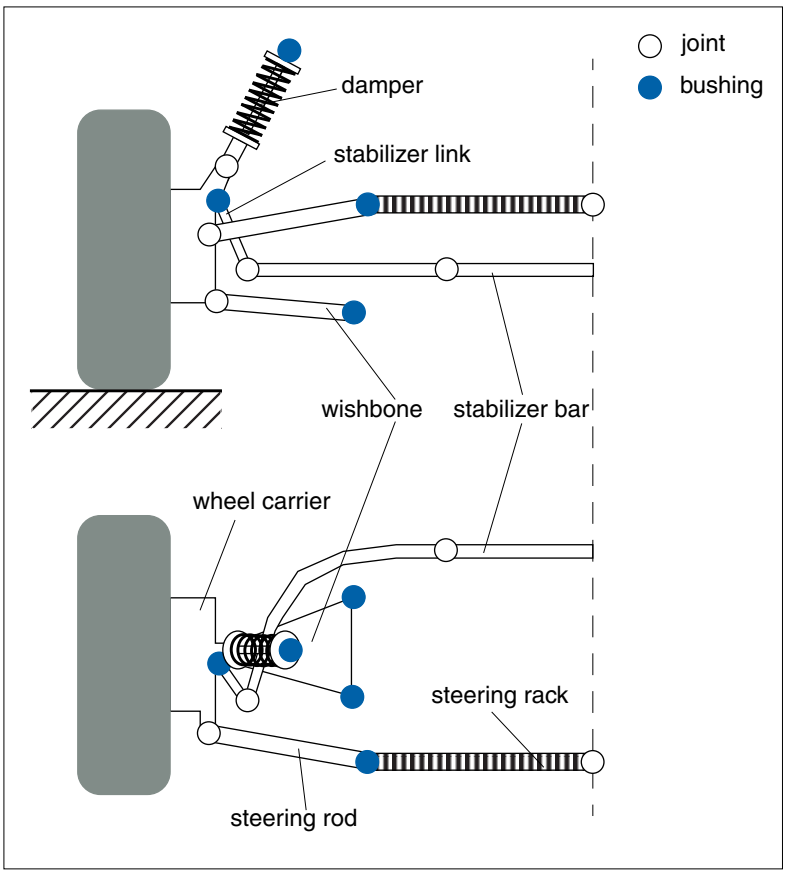


Figure 12.17: McPherson MBS suspension

For the selection of this MBS suspension model the *KindStr* ***McPherson*** must be set.

Example `FileIdent = CarMaker-SuspKnC-McPherson 1`

Bodies

The mechanical model of the McPherson suspension consists of 15 bodies: two wheels, two wheel carriers, two wishbones, two dampers, two stabilizer links, two stabilizer bar parts, two steering rods and one steering rack.

The left and right wheel bodies (position, mass and inertia) are defined in the vehicle dataset. The left and right wheel carrier bodies are described in the section 12.6.2 'General Parameters'.

Please do not confuse the wheel carrier body in the suspension dataset with the wheel carrier body in the vehicle dataset. The latter corresponds to a generalized body including also parts of the wishbone, damper, stabilizer link and steering rod besides the real wheel carrier. The MBS suspension model calculates the mass and inertia of the generalized wheel carrier body and checks if the values differ from the values defined in the vehicle dataset. A warning message informs about the expected values if the difference is large enough.

WishLowL.mass = *mass*
WishLowR.mass = *mass*
WishLowL.I = *Ixx Iyy Izz*
WishLowR.I = *Ixx Iyy Izz*

Specifies the left and right lower wishbone body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

DamperL.mass = *mass*
DamperR.mass = *mass*
DamperL.I = *Ixx Iyy Izz*
DamperR.I = *Ixx Iyy Izz*

Specifies the left and right damper body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.
The coil spring mass and inertia can be added partly to the wheel carrier and the damper bodies.

SteerRodL.mass = *mass*
SteerRodR.mass = *mass*
SteerRodL.I = *Ixx Iyy Izz*
SteerRodR.I = *Ixx Iyy Izz*

Specifies the left and right steering rod body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

SteerRack.mass = *mass*
SteerRack.I = *Ixx Iyy Izz*

Specifies the steering rack body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

StabiBar.mass = *mass*
StabiBar.I = *Ixx Iyy Izz*

Specifies the stabilizer bar body mass [kg] and inertia (main diagonal elements only) [kgm^2]. Internally the mass and inertia is divided into two body parts. The center of mass for both parts is calculated internally on the basis of bushing and joint positions.

StabiLinkL.mass = *mass*
StabiLinkR.mass = *mass*
StabiLinkL.I = *Ixx Iyy Izz*
StabiLinkR.I = *Ixx Iyy Izz*

Specifies the left and right stabilizer link body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

Joints and degrees of freedom

The McPherson suspension has 31 degrees of freedom from with 1 DOF for the movement of the steering rack travel is imposed by the steering model. The following table describes the suspension joints in detail which are illustrated in the Figure 12.18:

Joint	DOF	Relative movement
1	1	translation of damper body in direction of thrust axis relative to the wheel carrier body
2	3	3 rotations of wishbone body relative to the wheel carrier body with rotation sequence ZYX
3	2	2 rotations of steering rod body relative to the wheel carrier body with rotation sequence ZX
4	2	2 rotations of stabilizer link body relative to the stabilizer bar body with rotation sequence YX
5	1	rotation of stabilizer bar body relative to the vehicle chassis around the y-axis
6	1	translation of steering rack body relative to the vehicle chassis along the y-axis
7	6	3 translations and 3 rotations of the wheel carrier body relative to the vehicle chassis with rotation sequence ZXY

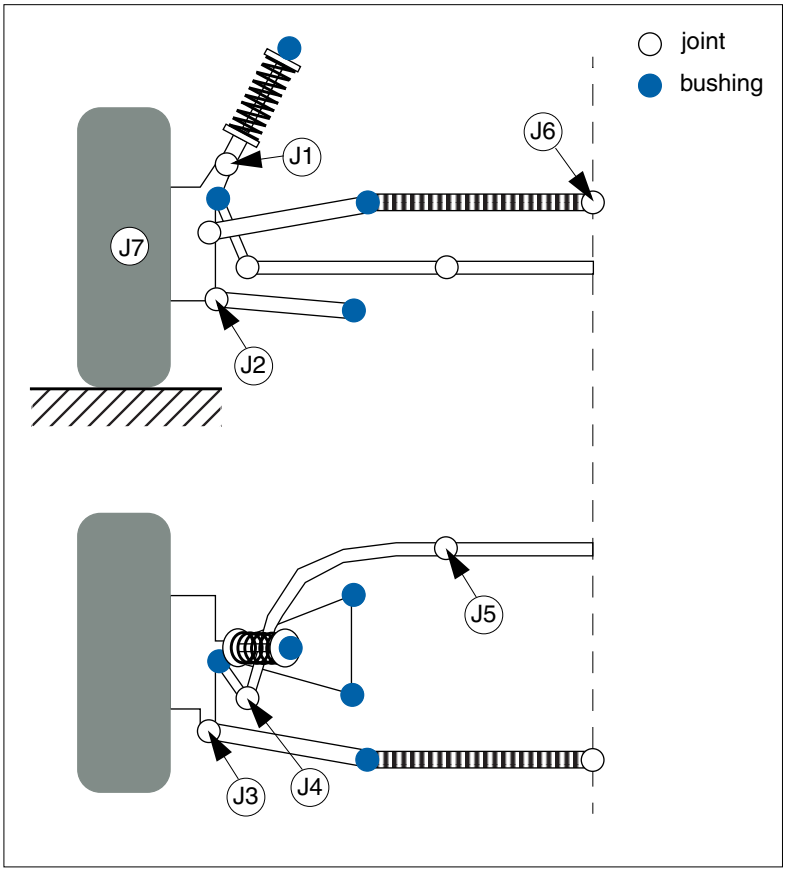


Figure 12.18: McPherson suspension joints

The position of the joint *J1* from the Figure 12.18 between the wheel carrier and the damper body is calculated internally on the basis of the upper and lower damper points.

CarrierL.Joint2Wish.pos =	***x***	***y***	***z***
CarrierR.Joint2Wish.pos =	***x***	***y***	***z***

Specifies the position of the joint *J2* from the Figure 12.18 between the wheel carrier and wishbone body expressed in `FrSusp` coordinates. Unit: [m].

CarrierL.Joint2StrRod.pos =	***x***	***y***	***z***
CarrierR.Joint2StrRod.pos =	***x***	***y***	***z***

Specifies the position of the joint *J3* from the Figure 12.18 between the wheel carrier and steering rod body expressed in `FrSusp` coordinates. Unit: [m].

StabiBarL.Joint2StLink.pos =	***x***	***y***	***z***
StabiBarR.Joint2StLink.pos =	***x***	***y***	***z***

Specifies the position of the joint *J4* from the Figure 12.18 between the stabilizer bar and stabilizer link body expressed in `FrSusp` coordinates. Unit: [m].

Subframe.Joint2StBarL.pos =	*x*	*y*	*z*
Subframe.Joint2StBarR.pos =	*x*	*y*	*z*

Specifies the position of the joint *J5* from the Figure 12.18 between the stabilizer bar and vehicle chassis expressed in `FrSusp` coordinates. Unit: [m].

The position of the joint *J6* from the Figure 12.18 between the vehicle chassis and the steering rack is calculated internally on the basis of both bushing positions on the steering rack.

The position of the joint *J7* from the Figure 12.18 between the wheel carrier body and the frozen vehicle chassis is located in the wheel carrier center of mass.

Force Elements

The position parameters of the upper and lower spring and damper points are described in section 12.6.2 'General Parameters'. The location of the push and pull buffers are supposed along the thrust axis of the damper.

Bushings

In the Figure 12.19, all bushings of the McPherson suspension are illustrated. The required *<BushingPre>* string for the bushing parameters from the section 12.6.2 'General Parameters' is listed below.

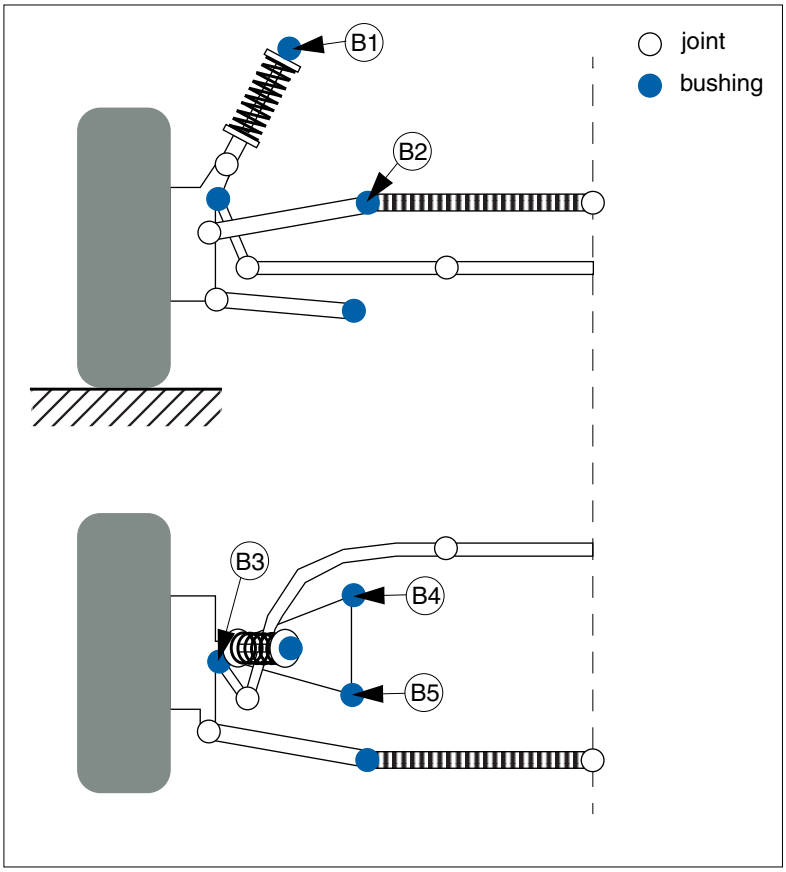


Figure 12.19: McPherson suspension bushings

Vehicle.Bush2DampL.*
Vehicle.Bush2DampR.*

Parameters of the left and right strut bushing *B1* from the Figure 12.19 between the damper body and vehicle chassis, beginning with the specified prefix.

SteerRodL.Bush2StrRack.*
SteerRodR.Bush2StrRack.*

Parameters of the left and right bushing *B2* from the Figure 12.19 between the steering rack and steering rod body, beginning with the specified prefix.

StabiLinkL.Bush2WC.*
StabiLinkR.Bush2WC.*

Parameters of the left and right bushing *B3* from the Figure 12.19 between the wheel carrier and stabilizer link body, beginning with the specified prefix.

WishLowL.BushF2Sub.*
WishLowR.BushF2Sub.*

Parameters of the left and right front bushing *B4* from the Figure 12.19 between the wishbone and vehicle chassis, beginning with the specified prefix.

WishLowL.BushR2Sub.*
WishLowR.BushR2Sub.*

Parameters of the left and right rear bushing *B5* from the Figure 12.19 between the wishbone and vehicle chassis, beginning with the specified prefix.

User Accessible Quantities

Please refer to section 24.8.2 'McPherson suspension' on page 765.

12.6.4 McPherson extended suspension

As shown in the Figure 12.20 the difference between the McPherson and McPherson extended suspension model are the two additional bodies for the steering box and the subframe. With these two additional bodies the number of degrees of freedom is extended by 11 further generalized coordinates, which are specified in detail below.

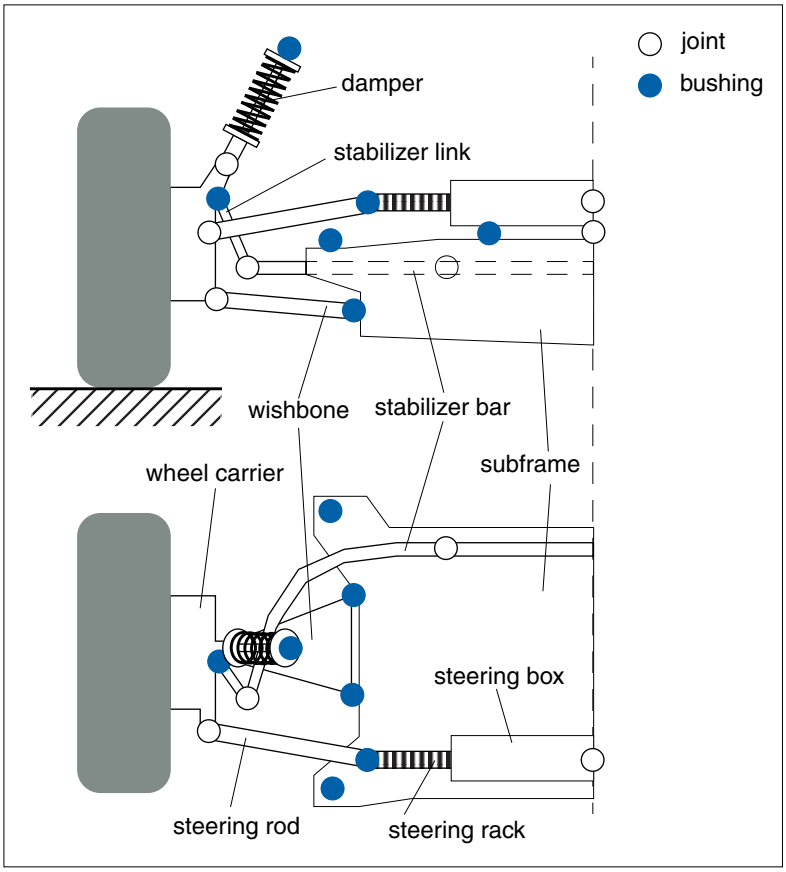


Figure 12.20: McPherson extended MBS suspension

For the selection of this suspension model the *KindStr* ***McPhersonExtend*** must be set.

Example `FileIdent = CarMaker-SuspKnC-McPhersonExtend 1`

Bodies

The mechanical model of the McPherson extended suspension consists of 17 bodies: two wheels, two wheel carriers, two wishbones, two dampers, two stabilizer links, two stabilizer bar parts, one steering rack, one steering box and the subframe.

Below, only the two additional bodies steering box and subframe are described. For other bodies please refer to section 12.6.3 'McPherson suspension'.

Subframe.mass	=	***mass***		
Subframe.I	=	***Ixx***	***Iyy***	***Izz***

Specifies the subframe body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

SteerBox.mass =	***mass***		
SteerBox.I =	***Ixx***	***Iyy***	***Izz***

Specifies the steering box body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

Joints and degrees of freedom

The table below describes only the new or modified suspension joints from the extended model in detail which are illustrated in the Figure 12.19. For the joints from the figure which are not mentioned in the table please refer to section 12.6.3 'McPherson suspension'.

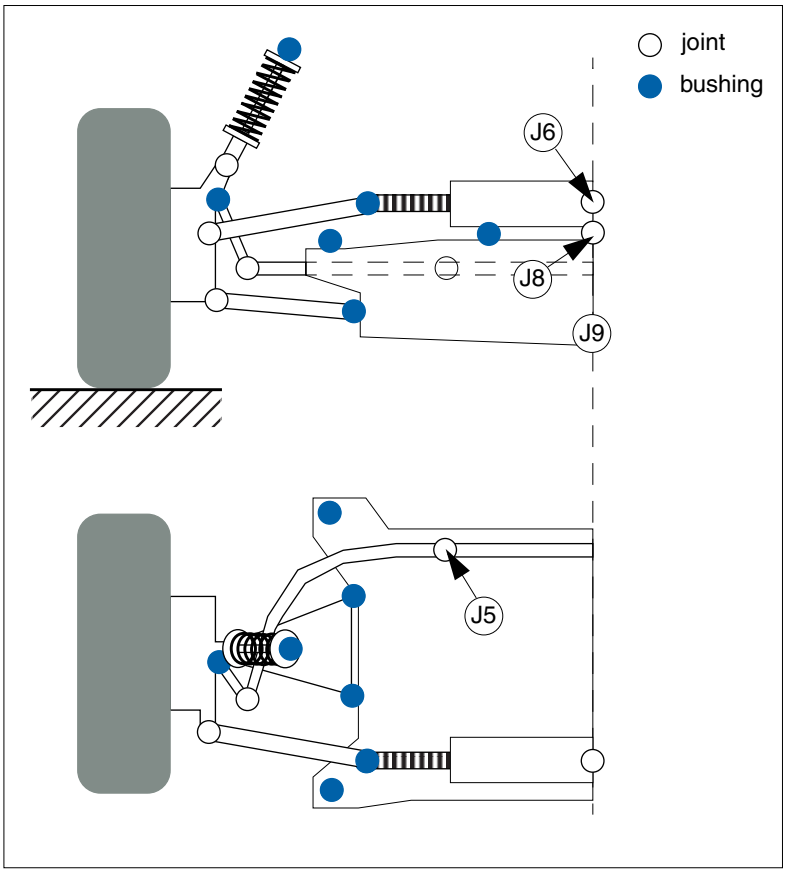


Figure 12.21: McPherson extended suspension joints

Joint	DOF	Relative movement
5	1	rotation of stabilizer bar body relative to subframe body around the y-axis
6	1	translation of steering rack body relative to steering box along the y-axis
8	5	3 translations and 2 rotations of the steering box relative to subframe body with rotation sequence ZX
9	6	3 translations and 3 rotations of the subframe body relative to vehicle chassis with rotation sequence ZYX

The position of the joint *J8* from the Figure 12.21 between the subframe and the steering box is calculated internally on the basis of the both bushing positions on the steering box.

The position of the joint *J9* from the Figure 12.21 between the subframe and the vehicle chassis is calculated internally on the basis of the four bushing positions on the subframe.

Bushings

With the two additional bodies in the McPherson extended suspension there come 6 further bushing components. In the Figure 12.22 all bushings of the McPherson extended suspension are illustrated. The required *<BushingPre>* string for the bushing parameters from the section 12.6.2 'General Parameters' is listed below.

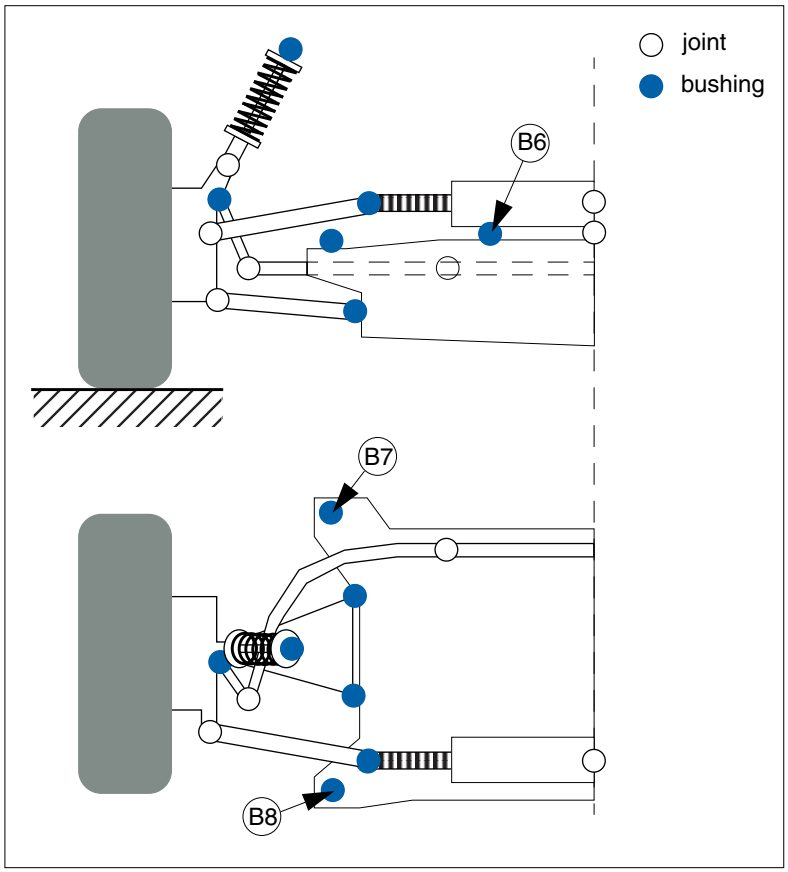


Figure 12.22: McPherson extended suspension bushings

At this point only the additional 6 bushings are listed. For detailed description of the other bushings please refer to section 12.6.3 'McPherson suspension'.

Subframe.BushL2StrBox.*
Subframe.BushR2StrBox.*

Parameters of the left and right bushing *B6* from the Figure 12.22 between the steering box body and the subframe body, beginning with the specified prefix.

Subframe.BushFL2Vhcl.*
Subframe.BushFR2Vhcl.*

Parameters of the left and right front bushing *B7* from the Figure 12.22 between the vehicle chassis and the subframe body, beginning with the specified prefix.

Subframe.BushRL2Vhcl.*
Subframe.BushRR2Vhcl.*

Parameters of the left and right rear bushing *B8* from the Figure 12.22 between the vehicle chassis and the subframe body, beginning with the specified prefix.

User Accessible Quantities

Please refer to section 24.8.3 'McPherson extended suspension' on page 766.

12.6.5 Fourlink suspension

The fourlink rear wheel suspension comprises one lower wishbone, one upper wishbone, one trailing arm and one steering rod per vehicle side that are mounted to the vehicle body with bushings (as shown in Figure 12.23). The lower wishbone, the upper wishbone, the trailing arm and the steering rod are externally connected to the wheel carrier via rotational joints.

A difference compared to a front wheel suspension model like the double wishbone suspension model is, that the fourth rod - the steering rod - is mounted directly to the vehicle body and the steering rack has been omitted.

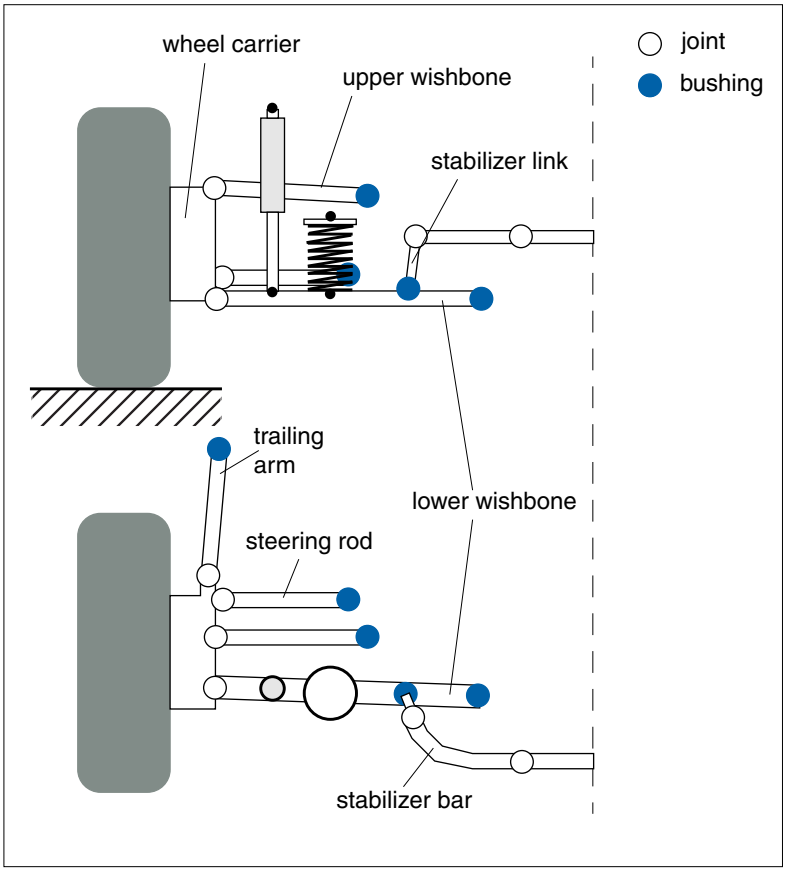


Figure 12.23: Fourlink MBS suspension

For the selection of this MBS suspension model the *KindStr* ***FourLink*** must be set.

Example `FileIdent = CarMaker-SuspKnC-FourLink 1`

Bodies

The mechanical model of the fourlink suspension consists of 16 bodies: two wheels, two wheel carriers, two upper wishbones, two lower wishbones, two trailing arms, two steering rods, two stabilizer links and two stabilizer bar parts.

The left and right wheel bodies (position, mass and inertia) are defined in the vehicle dataset. The left and right wheel carrier bodies are described in the section 12.6.2 'General Parameters'.

Please do not confuse the wheel carrier body in the suspension dataset with the wheel carrier body in the vehicle dataset. The latter corresponds to a generalized body including also parts of the wishbone, damper, stabilizer link and steering rod besides the real wheel carrier. The MBS suspension model calculates the mass and inertia of the generalized wheel carrier body and checks if the values differ from the values defined in the vehicle dataset. A warning message informs about the expected values if the difference is large enough.

WishLowL.mass =	***mass***		
WishLowR.mass =	***mass***		
WishLowL.I =	***Ixx***	***Iyy***	***Izz***
WishLowR.I =	***Ixx***	***Iyy***	***Izz***

Specifies the left and right lower wishbone body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

WishUppL.mass =	***mass***		
WishUppR.mass =	***mass***		
WishUppL.I =	***Ixx***	***Iyy***	***Izz***
WishUppR.I =	***Ixx***	***Iyy***	***Izz***

Specifies the left and right upper wishbone body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

TrailArmL.mass =	***mass***		
TrailArmR.mass =	***mass***		
TrailArmL.I =	***Ixx***	***Iyy***	***Izz***
TrailArmR.I =	***Ixx***	***Iyy***	***Izz***

Specifies the left and right trailing arm body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

SteerRodL.mass =	***mass***		
SteerRodR.mass =	***mass***		
SteerRodL.I =	***Ixx***	***Iyy***	***Izz***
SteerRodR.I =	***Ixx***	***Iyy***	***Izz***

Specifies the left and right steering rod body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

StabiBar.mass =	***mass***		
StabiBar.I =	***Ixx***	***Iyy***	***Izz***

Specifies the stabilizer bar body mass [kg] and inertia (main diagonal elements only) [kgm^2]. Internally the mass and inertia is divided into two body parts. The center of mass for both parts is calculated internally on the basis of bushing and joint positions.

StabiLinkL.mass =	***mass***		
StabiLinkR.mass =	***mass***		
StabiLinkL.I =	***Ixx***	***Iyy***	***Izz***
StabiLinkR.I =	***Ixx***	***Iyy***	***Izz***

Specifies the left and right stabilizer link body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

DamperL.mass =	***mass***		
DamperR.mass =	***mass***		
DamperL.I =	***Ixx***	***Iyy***	***Izz***
DamperR.I =	***Ixx***	***Iyy***	***Izz***

Specifies the left and right damper mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.
This information is only used for the calculation of the generalized vehicle wheel carrier mass and inertia.

SpringL.mass =	***mass***		
SpringR.mass =	***mass***		
SpringL.I =	***Ixx***	***Iyy***	***Izz***
SpringR.I =	***Ixx***	***Iyy***	***Izz***

Specifies the left and right coil spring mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.
This information is only used for the calculation of the generalized vehicle wheel carrier mass and inertia.

Joints and degrees of freedom

The fourlink suspension has 38 degrees of freedom which are illustrated in the Figure 12.24 and described in detail in the table below:

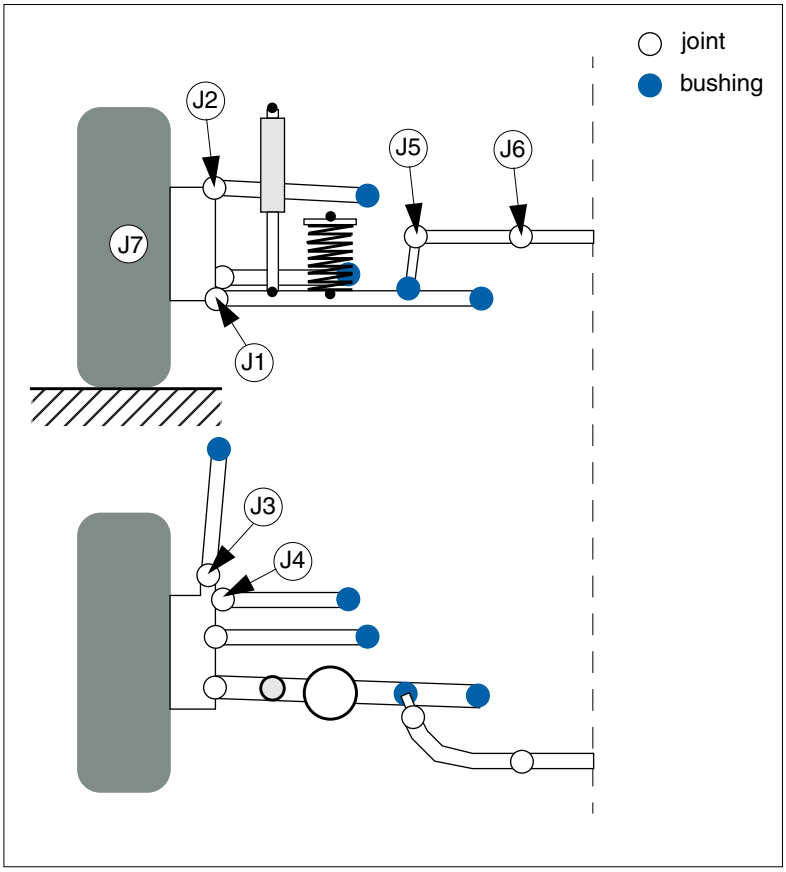


Figure 12.24: Fourlink MBS suspension joints

Joint	DOF	Relative movement
1	3	3 rotations of lower wishbone body relative to wheel carrier body with rotation sequence ZXY
2	3	3 rotations of upper wishbone body relative to wheel carrier body with rotation sequence ZXY
3	1	rotation of trailing arm body relative to wheel carrier body around z-axis (deflection of trailing arm)
4	3	3 rotations of steering rod body relative to wheel carrier body with rotation sequence ZXY
5	2	2 rotations of stabilizer link body relative to stabilizer bar body with rotation sequence YX
6	1	rotation of stabilizer bar body relative to vehicle chassis around the y-axis
7	6	3 translations and 3 rotations of the wheel carrier body relative to vehicle chassis with rotation sequence ZXY

CarrierL.Joint2WishLow.pos = *x* *y* *z*
CarrierR.Joint2WishLow.pos = *x* *y* *z*

Specifies the position of the joint *J1* from the Figure 12.24 between the wheel carrier and lower wishbone body expressed in `FrSusp` coordinates. Unit: [m].

CarrierL.Joint2WishUpp.pos = *x* *y* *z*
CarrierR.Joint2WishUpp.pos = *x* *y* *z*

Specifies the position of the joint *J2* from the Figure 12.24 between the wheel carrier and upper wishbone body expressed in `FrSusp` coordinates. Unit: [m].

CarrierL.Joint2TrailArm.pos = *x* *y* *z*
CarrierR.Joint2TrailArm.pos = *x* *y* *z*

Specifies the position of the joint *J3* from the Figure 12.24 between the wheel carrier and trailing arm body expressed in `FrSusp` coordinates. Unit: [m].

CarrierL.Joint2StrRod.pos = *x* *y* *z*
CarrierR.Joint2StrRod.pos = *x* *y* *z*

Specifies the position of the joint *J4* from the Figure 12.24 between the wheel carrier and steering rod body expressed in `FrSusp` coordinates. Unit: [m].

StabiBarL.Joint2StLink.pos = *x* *y* *z*
StabiBarR.Joint2StLink.pos = *x* *y* *z*

Specifies the position of the joint *J5* from the Figure 12.24 between the stabilizer bar and stabilizer link body expressed in `FrSusp` coordinates. Unit: [m].

Vehicle.Joint2StBarL.pos = *x* *y* *z*
Vehicle.Joint2StBarR.pos = *x* *y* *z*

Specifies the position of the joint *J6* from the Figure 12.24 between the stabilizer bar and vehicle chassis expressed in `FrSusp` coordinates. Unit: [m].

The position of the joint *J7* from the Figure 12.24 between the wheel carrier body and the frozen vehicle chassis is located in the wheel carrier center of mass.

Force Elements

The position parameters of the upper and lower spring and damper points are described in section 12.6.2 'General Parameters'. The location of the push and pull buffer are supposed along the thrust axis of the damper.

The damper, coil spring and the stabilizer link can be connected individually with the wheel carrier body or with the lower wishbone body:

Vehicle.SpringFrcOnWC = ***bool***

If 1, the coil spring is connected to the wheel carrier, else to lower wishbone (default).

Vehicle.DamperFrcOnWC = ***bool***

If 1, the damper is connected to the wheel carrier, else to lower wishbone (default).

StabiLink.BushFrcOnWC = ***bool***

If 1, the stabilizer link is connected to the wheel carrier, else to lower wishbone (default).

The trailing arm is a very thin flexible component, those deflection effects the wheel carrier compliance motion. For this reason the deflection of this body is considered in the mechanical model (*J3* in Figure 12.24) and is described by a linear rotational spring-damper element whose parameters are:

TrailArmL.cT = ***value***
TrailArmR.cT = ***value***

Specifies the trailing arm rotational spring characteristic value around the wheel carrier z-axis. This characteristic translates rotational compression around the z-axis to spring torque. Unit: [Nm/rad].

TrailArmL.kT = ***value***
TrailArmR.kT = ***value***

Specifies the trailing arm rotational damping characteristic value around the wheel carrier z-axis. This characteristic translates rotational compression velocity around the z-axis to spring torque. Unit: [Nms/rad].

Bushings

In the Figure 12.25 all bushings of the fourlink suspension are illustrated. The required *<BushingPre>* string for the bushing parameters from the section 12.6.2 'General Parameters' is listed below.

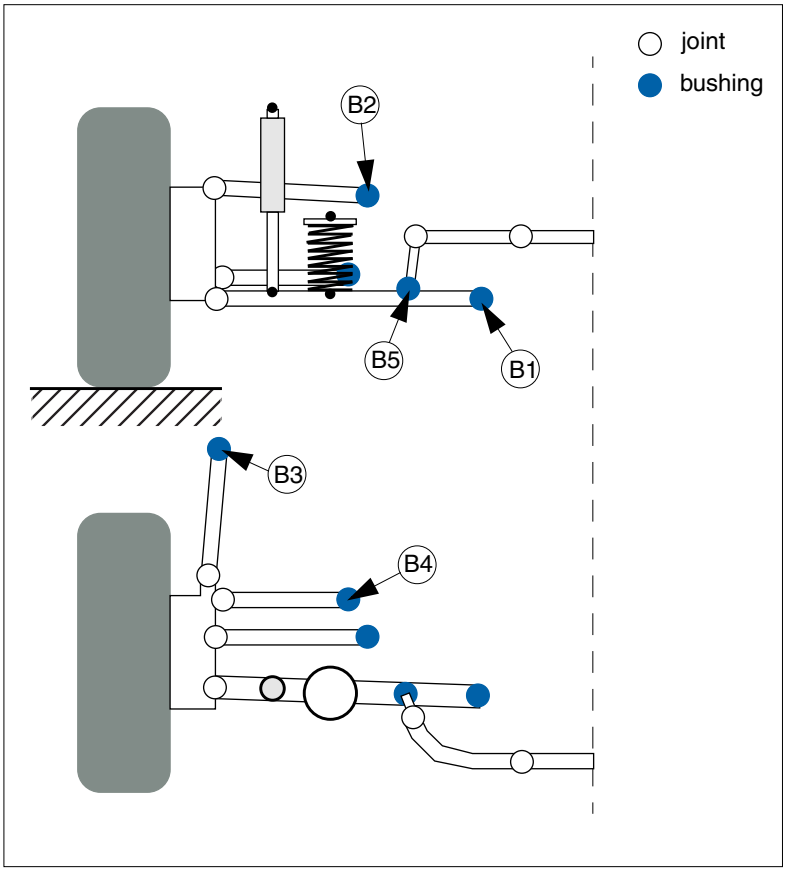


Figure 12.25: Fourlink MBS suspension bushings

WishLowL.Bush2Vhcl.*
WishLowR.Bush2Vhcl.*

Parameters of the left and right bushing *B1* from the Figure 12.25 between the lower wishbone body and the vehicle chassis, beginning with the specified prefix.

WishUppL.Bush2Vhcl.*
WishUppR.Bush2Vhcl.*

Parameters of the left and right bushing *B2* from the Figure 12.25 between the upper wishbone body and the vehicle chassis, beginning with the specified prefix.

TrailArmL.Bush2Vhcl.*
TrailArmR.Bush2Vhcl.*

Parameters of the left and right bushing *B3* from the Figure 12.25 between the trailing arm body and the vehicle chassis, beginning with the specified prefix.

SteerRodL.Bush2Vhcl.*
SteerRodR.Bush2Vhcl.*

Parameters of the left and right bushing *B4* from the Figure 12.25 between the steering rod body and the vehicle chassis, beginning with the specified prefix.

StabiLinkL.Bush.*
StabiLinkR.Bush.*

Parameters of the left and right bushing *B5* from the Figure 12.25 between the wheel carrier (or lower wishbone) and stabilizer link body, beginning with the specified prefix.

User Accessible Quantities

Please refer to section 24.8.4 'Fourlink suspension' on page 768.

12.6.6 Fourlink extended suspension

As shown in the Figure 12.26 the difference between the Fourlink and Fourlink extended suspension model is the additional body for the subframe. With these additional body the number of degrees of freedom is extended by 6 further generalized coordinates, which are specified in detail below.

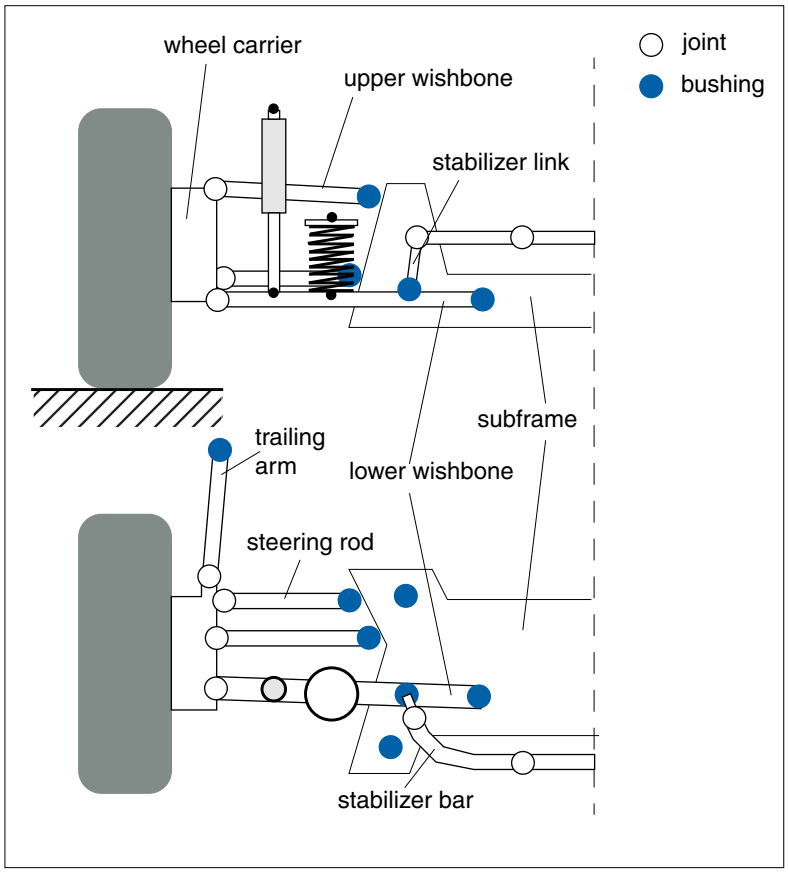


Figure 12.26: Fourlink extended MBS suspension

For the selection of this MBS suspension model the *KindStr* ***FourLinkExtend*** must be set.

Example `FileIdent = CarMaker-SuspKnC-FourLinkExtend 1`

Bodies

The mechanical model of the fourlink extended suspension consists of 17 bodies: two wheels, two wheel carriers, two upper wishbones, two lower wishbones, two trailing arms, two steering rods, two stabilizer links, two stabilizer bar parts and the subframe.

Below, only the additional body subframe is described. For other bodies please refer to section 12.6.5 'Fourlink suspension'.

Subframe.mass	=	***mass***		
Subframe.I	=	***Ixx***	***Iyy***	***Izz***

Specifies the subframe body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

Joints and degrees of freedom

The table below describes only the new or modified suspension joints from the extended model in detail which are illustrated in the Figure 12.27. For the joints from the figure which are not mentioned in the table please refer to section 12.6.5 'Fourlink suspension'.

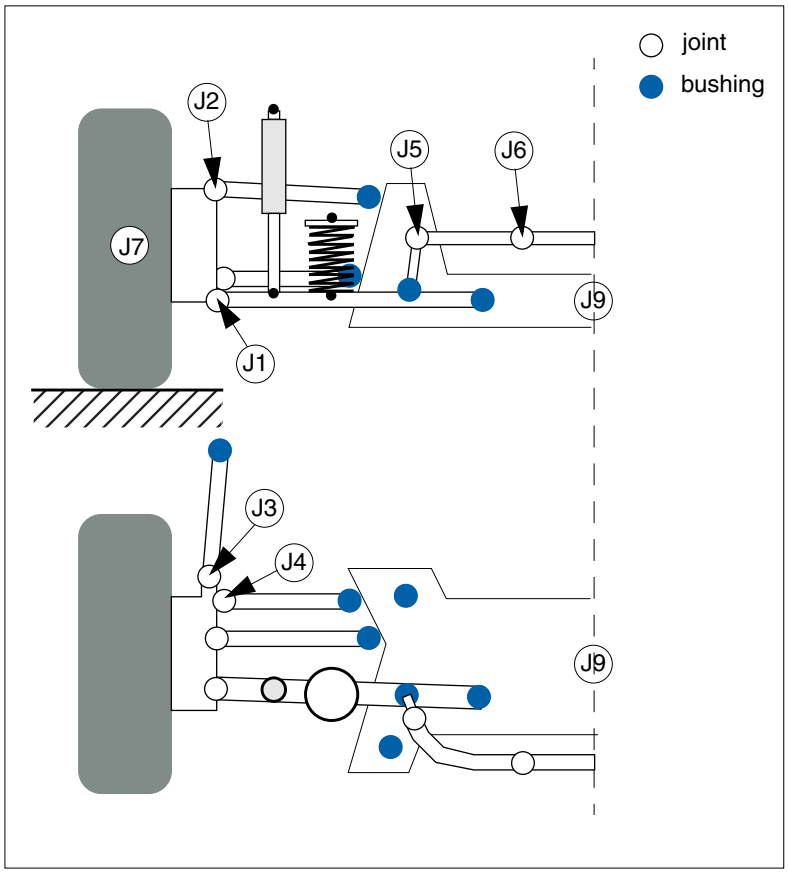


Figure 12.27: Fourlink extended suspension joints

Joint	DOF	Relative movement
9	6	3 translations and 3 rotations of the subframe body relative to vehicle chassis with rotation sequence ZYX

The position of the joint *J9* from the Figure 12.27 between the subframe and the vehicle chassis is calculated internally on the basis of the four bushing positions on the subframe.

Bushings

With the additional body in the Fourlink extended suspension there come 4 further bushing components. In the Figure 12.28 all bushings of the Fourlink extended suspension are illustrated. The required *<BushingPre>* string for the bushing parameters from the section 12.6.2 'General Parameters' is listed below.

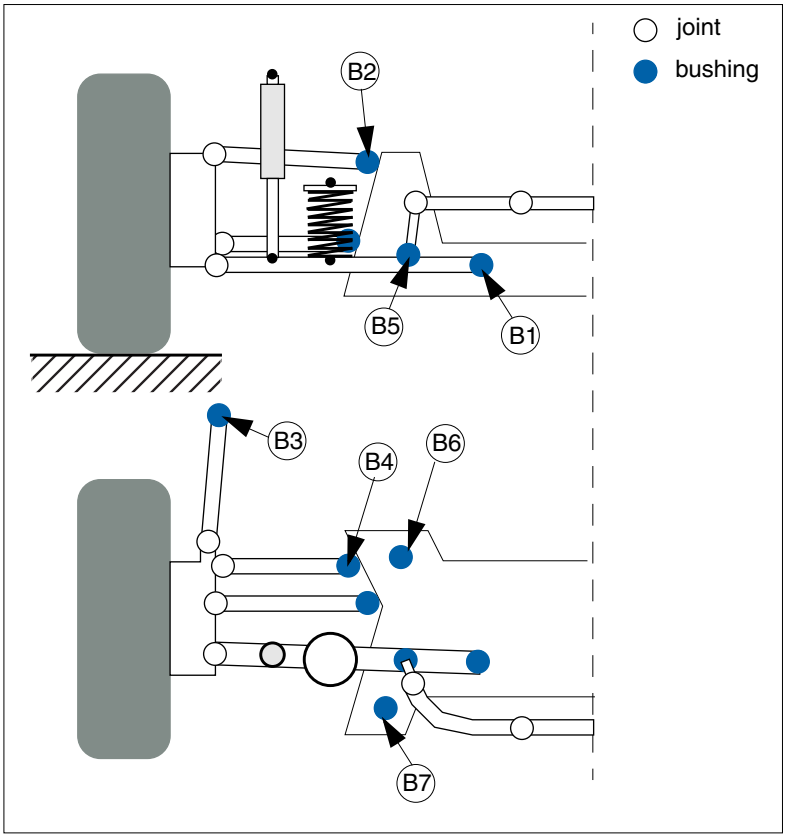


Figure 12.28: Fourlink extended suspension bushings

At this point only the additional 4 bushings are listed. For detailed description of the other bushings please refer to section 12.6.5 'Fourlink suspension'.

Subframe.BushFL2Vhcl.*
Subframe.BushFR2Vhcl.*

Parameters of the left and right front bushing *B6* from the Figure 12.28 between the vehicle chassis and the subframe body, beginning with the specified prefix.

Subframe.BushRL2Vhcl.*
Subframe.BushRR2Vhcl.*

Parameters of the left and right rear bushing *B7* from the Figure 12.28 between the vehicle chassis and the subframe body, beginning with the specified prefix.

User Accessible Quantities

Please refer to section 24.8.5 'Fourlink extended suspension' on page 770.

12.6.7 Twistbeam suspension

The modeling of a twistbeam rear axle MBS suspension is not as simple as the modeling of a McPherson or fourlink axle, where the construction elements like bodies, joints and bushings directly exist. The twistbeam rear axle has to be modeled using dummy bodies linked with torsional elements to simulate the elasticities of the torsion beam.

The twistbeam rear wheel suspension comprises one wheel carrier, one trailing arm per vehicle side and the torsion beam. The trailing arms are mounted on the vehicle body with bushings (as shown in Figure 12.29). The wheel carrier, the trailing arm and the torsion beam are connected via rotational joints.

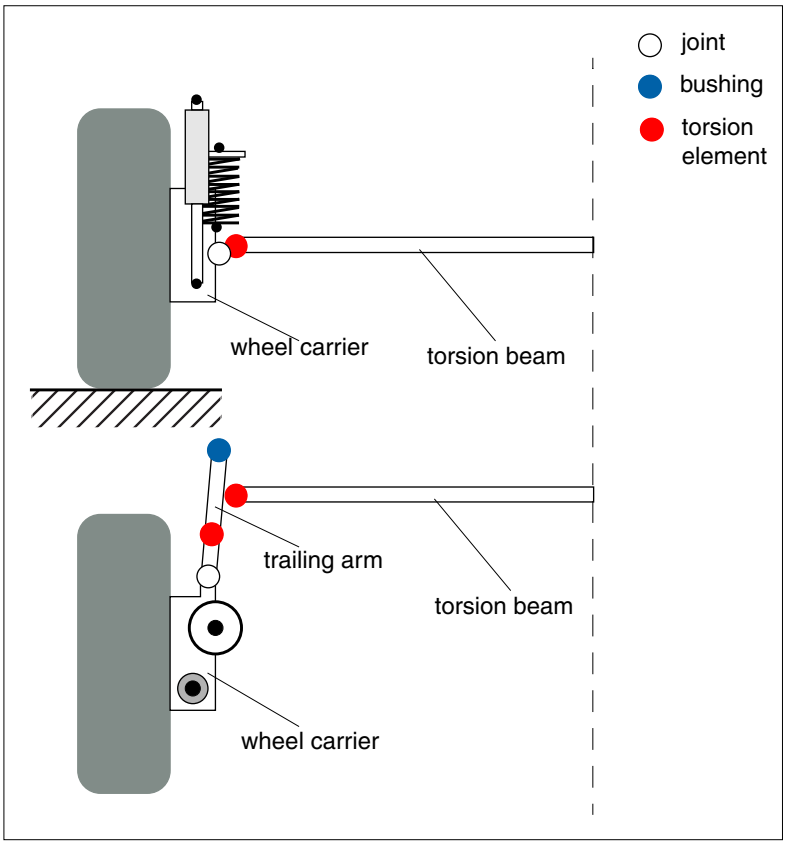


Figure 12.29: Twistbeam MBS suspension

In order to select this MBS suspension model the *KindStr* entry must be set to ***TwistBeam***.

Example `FileIdent = CarMaker-SuspKnC-TwistBeam 1`

Bodies

The mechanical model of the twistbeam suspension consists of seven bodies: two wheels, two wheel carriers, two trailing arms, and the torsion beam.

The left and right wheel bodies (position, mass and inertia) are defined in the vehicle dataset. The left and right wheel carrier bodies are described in section 12.6.2 'General Parameters'.

Please don't confuse the wheel carrier body in the suspension dataset with the wheel carrier body in the vehicle dataset. The latter corresponds to a generalized body including, besides the real wheel carrier, also parts of the trailing arm and damper. The MBS suspension model calculates the mass and inertia of the generalized wheel carrier body and checks if the values differ from the values defined in the vehicle dataset. A warning message informs about the expexted values if the difference is large enough.

TorsBeam.mass =	***mass***		
TorsBeam.I =	***Ixx***	***Iyy***	***Izz***

Specifies the torsion beam body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

TrailArmL.mass =	***mass***		
TrailArmR.mass =	***mass***		
TrailArmL.I =	***Ixx***	***Iyy***	***Izz***
TrailArmR.I =	***Ixx***	***Iyy***	***Izz***

Specifies the left and right trailing arm body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

DamperL.mass =	***mass***		
DamperR.mass =	***mass***		
DamperL.I =	***Ixx***	***Iyy***	***Izz***
DamperR.I =	***Ixx***	***Iyy***	***Izz***

Specifies the left and right damper mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.
This information is only used for calculation of the generalized vehicle wheel carrier mass and inertia.

SpringL.mass =	***mass***		
SpringR.mass =	***mass***		
SpringL.I =	***Ixx***	***Iyy***	***Izz***
SpringR.I =	***Ixx***	***Iyy***	***Izz***

Specifies the left and right coil spring mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.
This information is only used for calculation of the generalized vehicle wheel carrier mass and inertia.

Joints and degrees of freedom

The twistbeam suspension has 14 degrees of freedom which are illustrated in Figure 12.30 and described in detail in the table below:

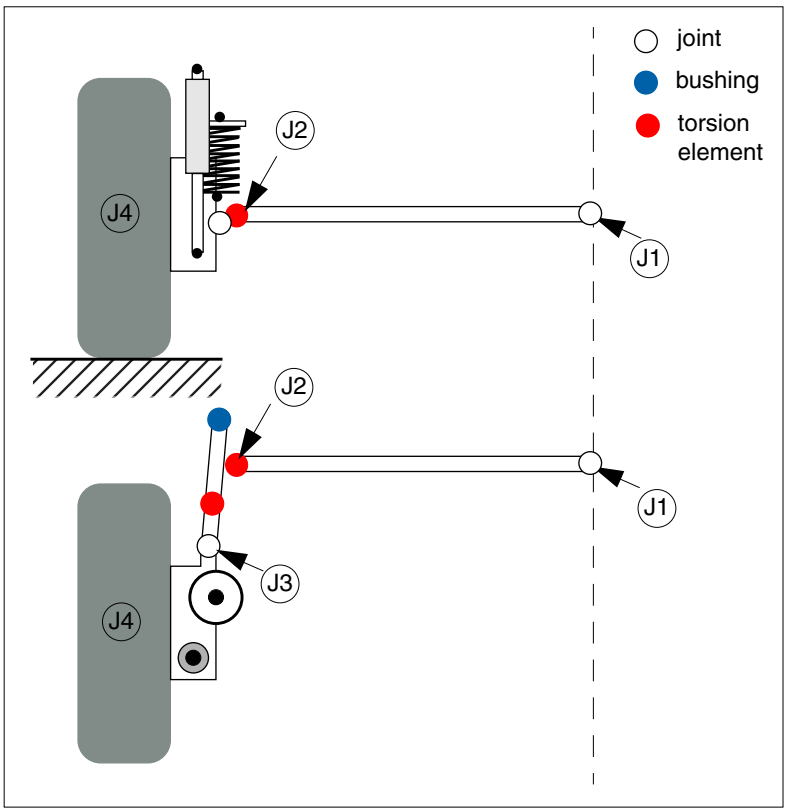


Figure 12.30: Twistbeam MBS suspension joints

Joint	DOF	Relative movement
1	6	3 translations and 3 rotations of the torsion beam body relative to the vehicle body with rotation sequence ZXY
2	3	3 rotations of the trailing arm body relative to the torsion beam with rotation sequence ZYX
3	1	1 rotation of the wheel carrier body relative to the trailing arm body around the z-axis (bending of the trailing arm)

CarrierL.Joint2TrailArm.pos = *x* *y* *z*
CarrierR.Joint2TrailArm.pos = *x* *y* *z*

Specifies the position of the virtual joint *J3* from Figure 12.30 between the wheel carrier and the trailing arm body expressed in `FrSusp` coordinates. Unit: [m].

TrailArmL.Joint2TorsBeam.pos = *x* *y* *z*
TrailArmR.Joint2TorsBeam.pos = *x* *y* *z*

Specifies the position of the virtual joint *J2* from Figure 12.30 between the trailing arm body and the torsion beam body expressed in `FrSusp` coordinates. Unit: [m].

The position of joint *J4* from Figure 12.30 between the wheel carrier body and the frozen vehicle chassis is located in the wheel carrier center of mass.

Force Elements

The position parameters of the upper and lower spring and damper points are described in section 12.6.2 'General Parameters'. The location of the push and pull buffer is supposed to be along the thrust axis of the damper.

The torsion beam is a flexible component; this bending effects the trailing arm compliance motion. For this reason the bending of this body is considered in the mechanical model (*J2* in Figure 12.30) and is described by linear rotational spring-damper elements with the following parameters:

TrailArmL.cT = ***cTx cTy cTz***
TrailArmR.cT = ***cTx cTy cTz***

Specifies the trailing arm rotational spring characteristic value around the torsion beam x-y-z-axis. This characteristic translates rotational compression around the x-y-z-axis to spring torque. Unit: [Nm/rad].

TrailArmL.kT = ***kTx kTy kTz***
TrailArmR.kT = ***kTx kTy kTz***

Specifies the trailing arm rotational damping characteristic value around the torsion beam x-y-z-axis. This characteristic translates rotational compression velocity around the x-y-z-axis to damping torque. Unit: [Nms/rad].

The trailing arm is a flexible component, this bending effects the wheel carrier compliance motion. For this reason the bending of this body is considered in the mechanical model (*J3* in Figure 12.30) and is described by a linear rotational spring-damper element with the following parameters:

CarrierL.cT = ***cTz***
CarrierR.cT = ***cTz***

Specifies the trailing arm rotational spring characteristic value around the wheel carrier z-axis. This characteristic translates rotational compression around the z-axis to spring torque. Unit: [Nm/rad].

CarrierL.kT = ***kTz***
CarrierR.kT = ***kTz***

Specifies the trailing arm rotational damping characteristic value around the wheel carrier z-axis. This characteristic translates rotational compression velocity around the z-axis to damping torque. Unit: [Nms/rad].

Bushings

In Figure 12.31 all bushings of the twistbeam suspension are illustrated. The required *<BushingPre>* string for the bushing parameters from the section 12.6.2 'General Parameters' is listed below.

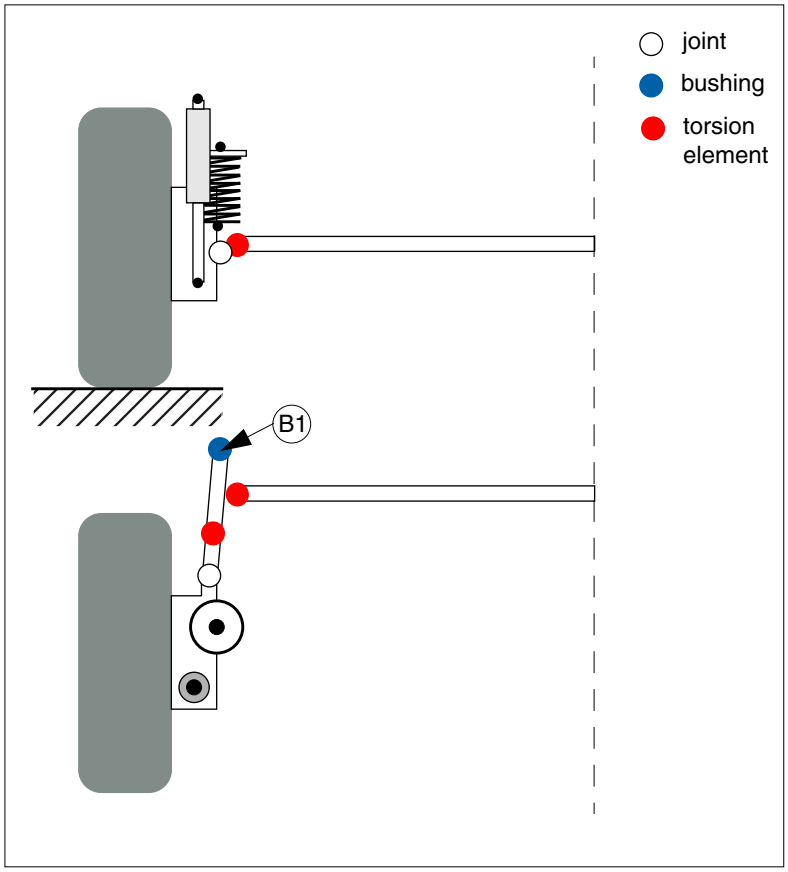


Figure 12.31: Twistbeam MBS suspension bushings

TrailArmL.Bush2Vhcl.*
TrailArmR.Bush2Vhcl.*

Parameters of the left and right bushing *B1* from Figure 12.31 between the trailing arm body and the vehicle chassis, beginning with the specified prefix.

User Accessible Quantities

Please refer to section 24.8.6 'Twistbeam suspension' on page 771.

12.6.8 Twistbeam extended suspension

The difference between the Twistbeam and Twisbeam extended suspension model is the additional elasticity of the torsion beam. The torsion element between the trailing arm and the twist beam is replaced by the functionality of a bushing.

In order to select this MBS suspension model the *KindStr* entry must be set to ***TwistBeamExtend***.

Example `FileIdent = CarMaker-SuspKnC-TwistBeamExtend 1`

Bushings

In Figure 12.32 all bushings of the twistbeam suspension are illustrated. The required *<BushingPre>* string for the bushing parameters from the section 12.6.2 'General Parameters' is listed below.

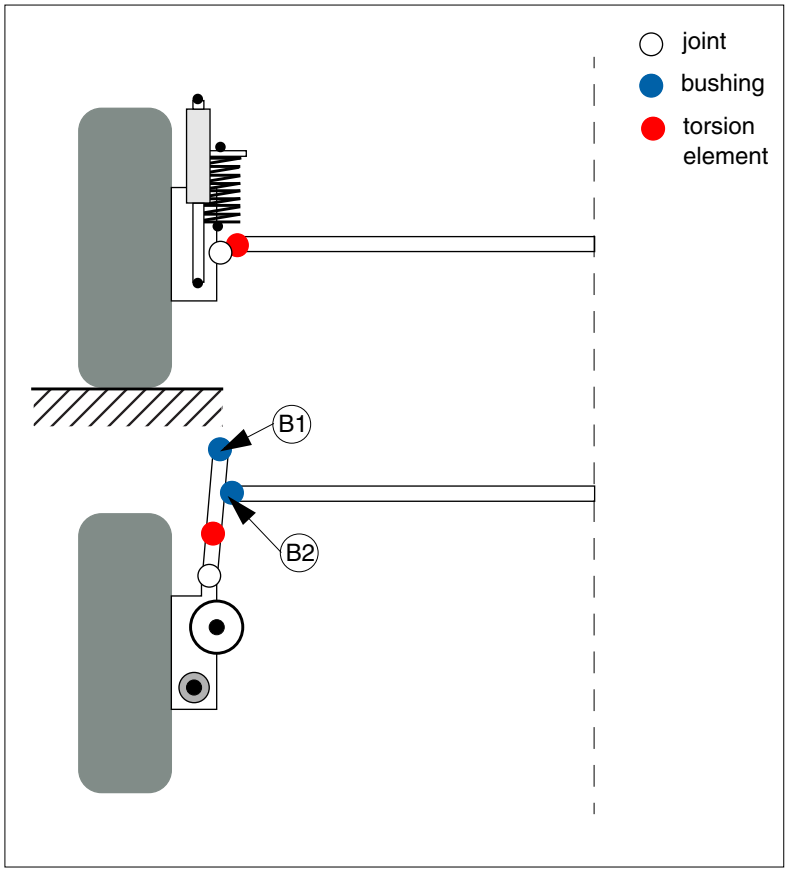


Figure 12.32: Twistbeam extended MBS suspension bushings

TrailArmL.Bush2Vhcl.*
TrailArmR.Bush2Vhcl.*

Parameters of the left and right bushing *B1* from Figure 12.32 between the trailing arm body and the vehicle chassis, beginning with the specified prefix.

TrailArmL.Bush2TorsBeam.*
TrailArmR.Bush2TorsBeam.*

Parameters of the left and right bushing *B2* from Figure 12.32 between the trailing arm body and the torsion beam, beginning with the specified prefix.

User Accessible Quantities

Please refer to section 24.8.7 'Twistbeam extended suspension' on page 772.

12.6.9 Double wishbone suspension

The double wishbone wheel suspension consists of two wishbones per vehicle side that are pivot mounted on the vehicle body. The wishbones are externally connected to the wheel carrier via ball joints or swivel bearings. Figure 12.33 is a schematic illustration of a double wishbone front suspension on the left vehicle side.

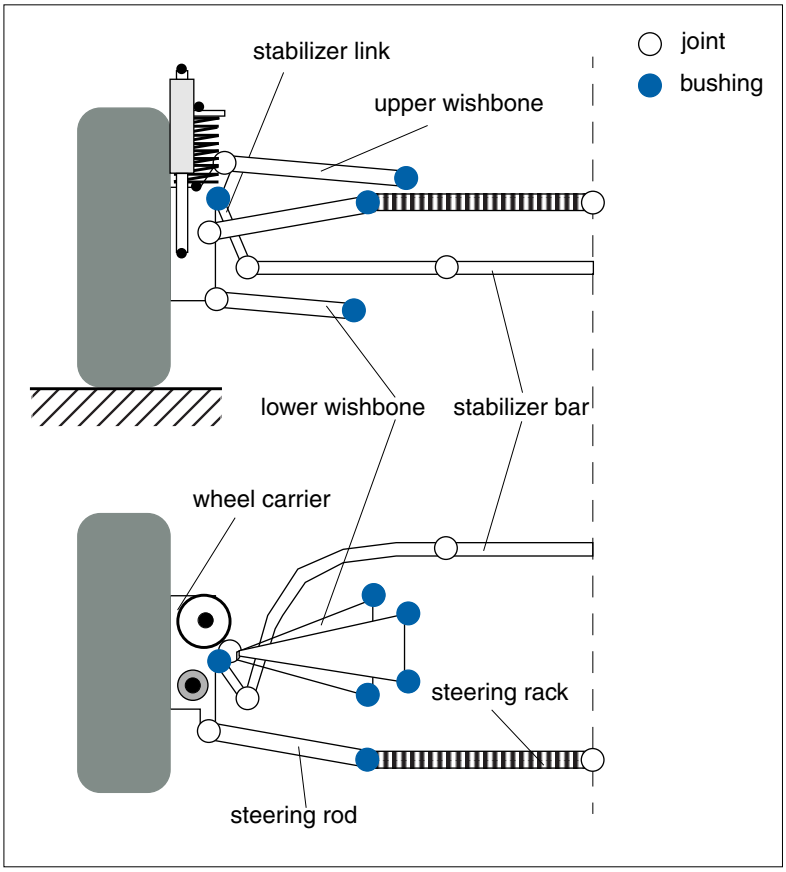


Figure 12.33: Double wishbone front MBS suspension

For the selection of this MBS suspension model the *KindStr* ***DoubleWishbone*** must be set.

Example `FileIdent = CarMaker-SuspKnC-DoubleWishbone 1`

General

The double wishbone suspension can be used as a front axle with steering system as well as a rear axle without steering system. In the case of a front axle *qComp* and *qSteer* are the relevant degrees of freedom, in the case of a rear axle *qComp* and *qCompOpp* are used. Figure 12.34 is a schematic illustration of a double wishbone rear suspension on the left vehicle side.

WithSteering = ***bool***

If set to 1 (default), the double wishbone suspension includes a steering system.

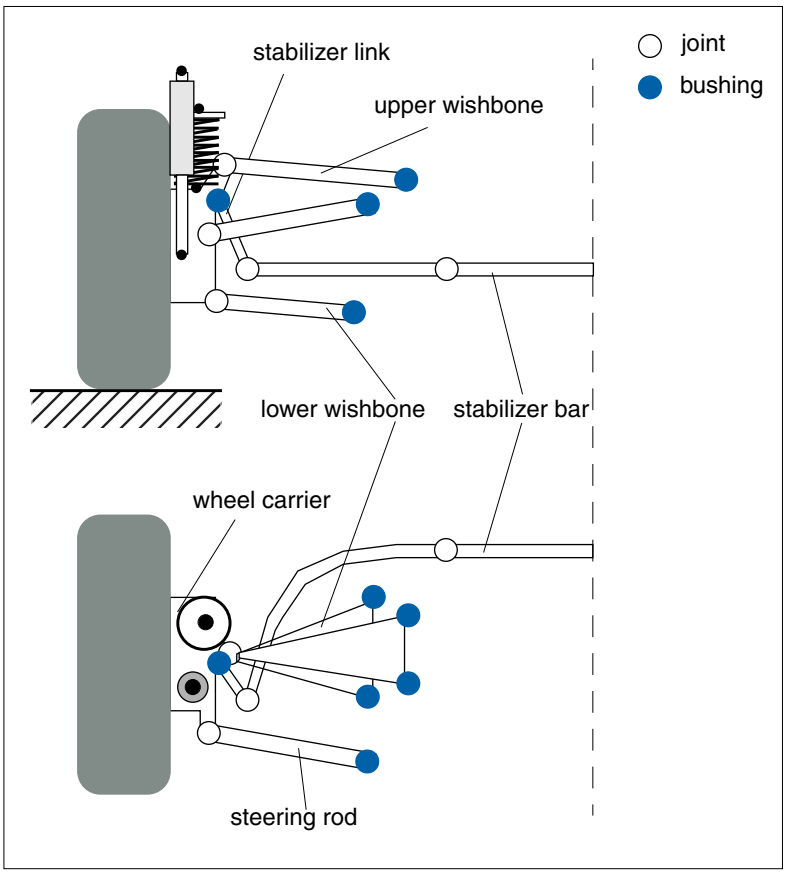


Figure 12.34: Double wishbone rear MBS suspension

Bodies

The mechanical model of the double wishbone suspension consists of 15 bodies: two wheels, two wheel carriers, two lower wishbones, two upper wishbones, two stabilizer links, two stabilizer bar parts, two steering rods and one steering rack.

The left and right wheel bodies (position, mass and inertia) are defined in the vehicle dataset. The left and right wheel carrier bodies are described in section 12.6.2 'General Parameters'.

Please do not confuse the wheel carrier body in the suspension dataset with the wheel carrier body in the vehicle dataset. The latter corresponds to a generalized body including also parts of the lower and upper wishbone, spring, damper, stabilizer link and steering rod besides the real wheel carrier. The MBS suspension model calculates the mass and inertia of the generalized wheel carrier body and checks if the values differ from the values defined in the vehicle dataset. A warning message informs about the expected values if the difference is large enough.

WishLowL.mass =	***mass***		
WishLowR.mass =	***mass***		
WishLowL.I =	***Ixx***	***Iyy***	***Izz***
WishLowR.I =	***Ixx***	***Iyy***	***Izz***

Specifies the left and right lower wishbone body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

WishUppL.mass =	***mass***		
WishUppR.mass =	***mass***		
WishUppL.I =	***Ixx***	***Iyy***	***Izz***
WishUppR.I =	***Ixx***	***Iyy***	***Izz***

Specifies the left and right upper wishbone body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

DamperL.mass =	***mass***		
DamperR.mass =	***mass***		
DamperL.I =	***Ixx***	***Iyy***	***Izz***
DamperR.I =	***Ixx***	***Iyy***	***Izz***

Specifies the left and right damper mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of joint positions.
This information is only used for calculation of the generalized vehicle wheel carrier mass and inertia.

SpringL.mass =	***mass***		
SpringR.mass =	***mass***		
SpringL.I =	***Ixx***	***Iyy***	***Izz***
SpringR.I =	***Ixx***	***Iyy***	***Izz***

Specifies the left and right coil spring mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of joint positions.
This information is only used for calculation of the generalized vehicle wheel carrier mass and inertia.

SteerRodL.mass = *mass*
SteerRodR.mass = *mass*
SteerRodL.I = *Ixx Iyy Izz*
SteerRodR.I = *Ixx Iyy Izz*

Specifies the left and right steering rod body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

SteerRack.mass = *mass*
SteerRack.I = *Ixx Iyy Izz*

Specifies the steering rack body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

StabiBar.mass = *mass*
StabiBar.I = *Ixx Iyy Izz*

Specifies the stabilizer bar body mass [kg] and inertia (main diagonal elements only) [kgm^2]. Internally the mass and inertia is divided into two body parts. The center of mass for both parts is calculated internally on the basis of bushing and joint positions.

StabiLinkL.mass = *mass*
StabiLinkR.mass = *mass*
StabiLinkL.I = *Ixx Iyy Izz*
StabiLinkR.I = *Ixx Iyy Izz*

Specifies the left and right stabilizer link body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

Joints and degrees of freedom

The double wishbone suspension has 35 degrees of freedom (DOF) with 1 DOF for the movement of the steering rack travel being imposed by the steering model. The following table describes in detail the suspension joints which are illustrated in the Figure 12.35:

Joint	DOF	Relative movement
1	3	3 rotations of lower wishbone body relative to the wheel carrier body with rotation sequence ZYX
2	3	3 rotations of upper wishbone body relative to the wheel carrier body with rotation sequence ZYX
3	2	2 rotations of steering rod body relative to the wheel carrier body with rotation sequence ZX

Joint	DOF	Relative movement
4	2	2 rotations of stabilizer link body relative to the stabilizer bar body with rotation sequence YX
5	1	rotation of stabilizer bar body relative to the vehicle chassis around the y-axis
6	1	translation of steering rack body relative to the vehicle chassis along the y-axis
7	6	3 translations and 3 rotations of the wheel carrier body relative to the vehicle chassis with rotation sequence ZXY

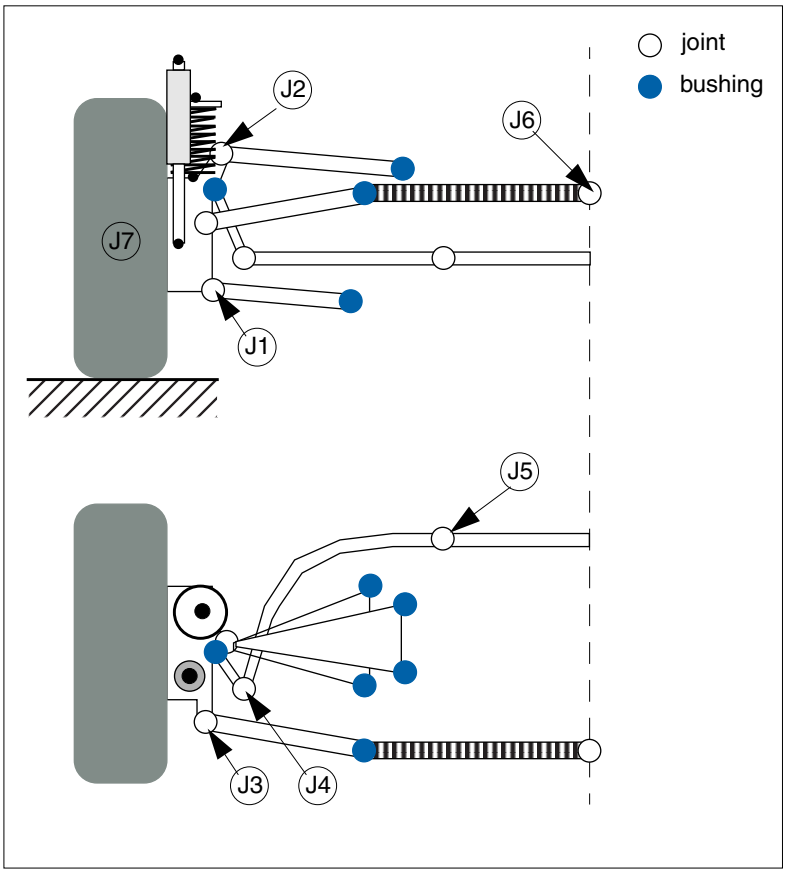


Figure 12.35: Double wishbone suspension joints

CarrierL.Joint2WishLow.pos = *x* *y* *z*
CarrierR.Joint2WishLow.pos = *x* *y* *z*

Specifies the position of the joint *J1* from the Figure 12.35 between the wheel carrier and the lower wishbone body expressed in `FrSusp` coordinates. Unit: [m].

CarrierL.Joint2WishUpp.pos = *x* *y* *z*
CarrierR.Joint2WishUpp.pos = *x* *y* *z*

Specifies the position of the joint *J2* from the Figure 12.35 between the wheel carrier and the upper wishbone body expressed in `FrSusp` coordinates. Unit: [m].

CarrierL.Joint2StrRod.pos = *x* *y* *z*
CarrierR.Joint2StrRod.pos = *x* *y* *z*

Specifies the position of the joint *J3* from the Figure 12.35 between the wheel carrier and steering rod body expressed in `FrSusp` coordinates. Unit: [m].

StabiBarL.Joint2StLink.pos = *x* *y* *z*
StabiBarR.Joint2StLink.pos = *x* *y* *z*

Specifies the position of the joint *J4* from the Figure 12.35 between the stabilizer bar and stabilizer link body expressed in `FrSusp` coordinates. Unit: [m].

Vehicle.Joint2StBarL.pos = *x* *y* *z*
Vehicle.Joint2StBarR.pos = *x* *y* *z*

Specifies the position of the joint *J5* from the Figure 12.35 between the stabilizer bar and vehicle chassis expressed in `FrSusp` coordinates. Unit: [m].

The position of the joint *J6* from the Figure 12.35 between the vehicle chassis and the steering rack is calculated internally on the basis of both bushing positions on the steering rack.

The position of the joint *J7* from the Figure 12.35 between the wheel carrier body and the frozen vehicle chassis is located in the wheel carrier center of mass.

Force Elements

The position parameters of the upper and lower spring and damper points are described in section 12.6.2 'General Parameters'. The location of the push and pull buffers is supposed to be along the thrust axis of the damper.

The damper, coil spring and the stabilizer link can be connected individually with the wheel carrier body or with the lower wishbone body:

Vehicle.SpringFrcOnWC = ***bool***

If 1, the coil spring is connected to the wheel carrier, else to the lower wishbone (default).

Vehicle.DamperFrcOnWC = ***bool***

If 1, the damper is connected to the wheel carrier, else to the lower wishbone (default).

StabiLink.BushFrcOnWC = ***bool***

If 1, the stabilizer link is connected to the wheel carrier, else to the lower wishbone (default).

Bushings

In the Figure 12.36, all bushings of the double wishbone suspension are illustrated. The required *<BushingPre>* string for the bushing parameters from the section 12.6.2 'General Parameters' is listed below.

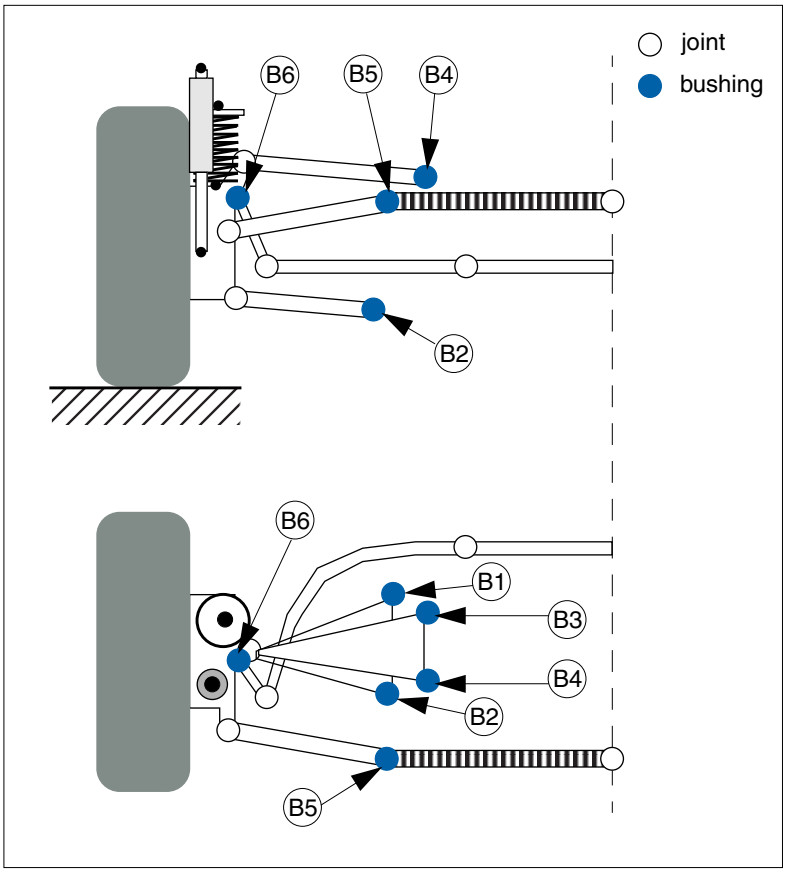


Figure 12.36: Double wishbone suspension bushings

WishLowL.BushF2Vhcl.*
WishLowR.BushF2Vhcl.*

Parameters of the left and right front bushing *B1* from Figure 12.36 between the lower wishbone and vehicle chassis, beginning with the specified prefix.

WishLowL.BushR2Vhcl.*
WishLowR.BushR2Vhcl.*

Parameters of the left and right rear bushing *B2* from Figure 12.36 between the lower wishbone and vehicle chassis, beginning with the specified prefix.

WishUppL.BushF2Vhcl.*
WishUppR.BushF2Vhcl.*

Parameters of the left and right front bushing *B3* from Figure 12.36 between the upper wishbone and vehicle chassis, beginning with the specified prefix.

WishUppL.BushR2Vhcl.*
WishUppR.BushR2Vhcl.*

Parameters of the left and right rear bushing *B4* from Figure 12.36 between the upper wishbone and vehicle chassis, beginning with the specified prefix.

SteerRodL.Bush2StrRack.*
SteerRodR.Bush2StrRack.*

Parameters of the left and right bushing *B5* from Figure 12.36 between the steering rack and steering rod body, beginning with the specified prefix.

StabiLinkL.Bush2WC.*
StabiLinkR.Bush2WC.*

Parameters of the left and right bushing *B6* from Figure 12.36 between the wheel carrier and stabilizer link body, beginning with the specified prefix.

User Accessible Quantities

Please refer to section 24.8.8 'Double wishbone suspension' on page 773.

12.6.10 Double wishbone extended suspension

As shown in the Figure 12.37 the difference between the Double wishbone and Double wishbone extended suspension model are the two additional bodies for the steering box and the subframe. With these two additional bodies the number of degrees of freedom is extended by 11 further generalized coordinates, which are specified in detail below.

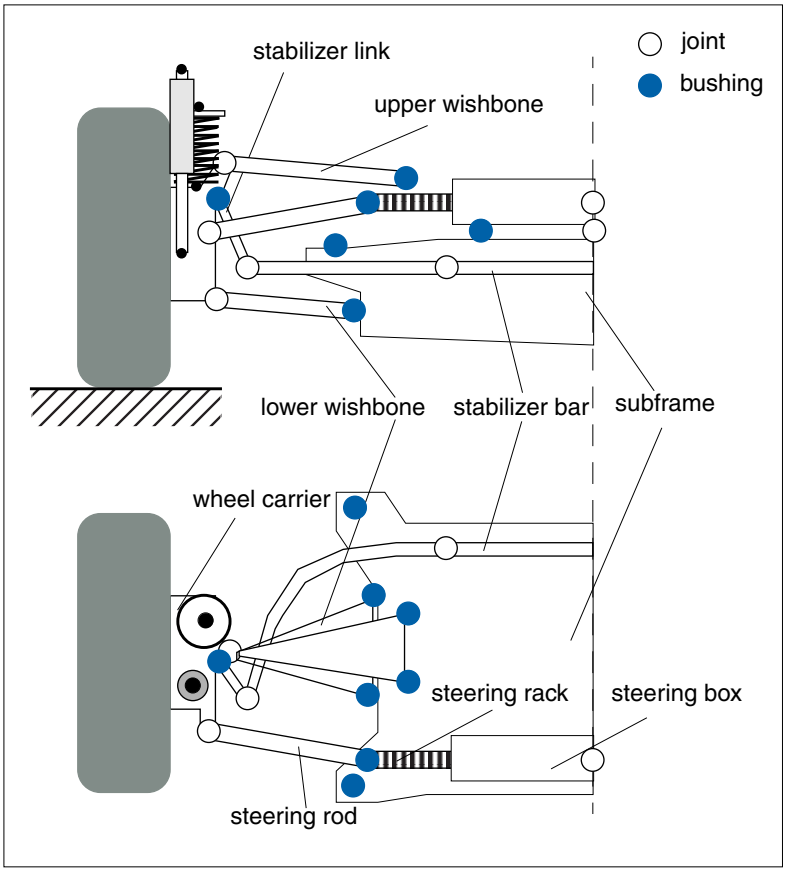


Figure 12.37: Double wishbone extended MBS suspension

For the selection of this MBS suspension model the *KindStr* ***DoubleWishboneExtend*** must be set.

Example `FileIdent = CarMaker-SuspKnC-DoubleWishboneExtend 1`

Bodies

The mechanical model of the Double wishbone extended suspension consists of 17 bodies: two wheels, two wheel carriers, two lower wishbones, two upper wishbones, two stabilizer links, two stabilizer bar parts, two steering rods, one steering rack, one steering box and the subframe.

Below, only the two additional bodies steering box and subframe are described. For other bodies please refer to section 12.6.9 'Double wishbone suspension'.

Subframe.mass = ***mass***
Subframe.I = ***Ixx Iyy Izz***

Specifies the subframe body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

SteerBox.mass = ***mass***
SteerBox.I = ***Ixx Iyy Izz***

Specifies the steering box body mass [kg] and inertia (main diagonal elements only) [kgm^2]. The center of mass is calculated internally on the basis of bushing and joint positions.

Joints and degrees of freedom

The table below describes only the new or modified suspension joints from the extended model in detail which are illustrated in the Figure 12.38. For the joints from the figure which are not mentioned in the table please refer to section 12.6.9 'Double wishbone suspension'.

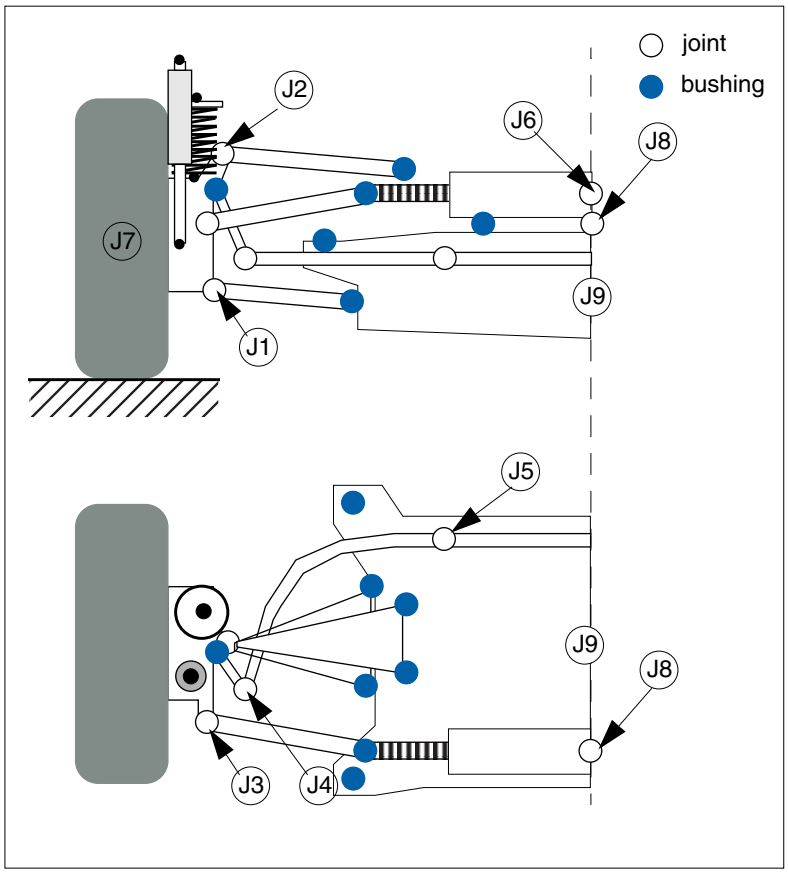


Figure 12.38: Double wishbone extended suspension joints

Joint	DOF	Relative movement
5	1	rotation of stabilizer bar body relative to subframe body around the y-axis
6	1	translation of steering rack body relative to steering box along the y-axis

Joint	DOF	Relative movement
8	5	3 translations and 2 rotations of the steering box relative to subframe body with rotation sequence ZX
9	6	3 translations and 3 rotations of the subframe body relative to vehicle chassis with rotation sequence ZYX

The position of the joint *J8* from the Figure 12.38 between the subframe and the steering box is calculated internally on the basis of the both bushing positions on the steering box.

The position of the joint *J9* from the Figure 12.38 between the subframe and the vehicle chassis is calculated internally on the basis of the four bushing positions on the subframe.

Bushings

With the two additional bodies in the Double wishbone extended suspension there come 6 further bushing components. In the Figure 12.39 all bushings of the Double wishbone extended suspension are illustrated. The required *<BushingPre>* string for the bushing parameters from the section 12.6.2 'General Parameters' is listed below.

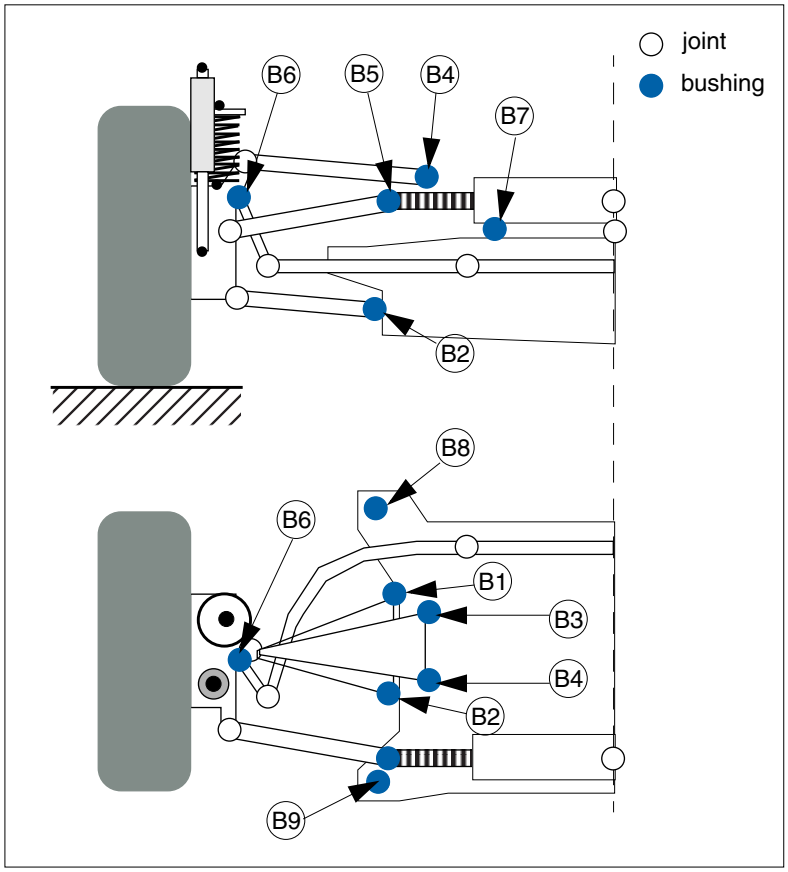


Figure 12.39: Double wishbone extended suspension bushings

At this point only the additional 6 bushings are listed. For detailed description of the other bushings please refer to section 12.6.9 'Double wishbone suspension'.

Subframe.BushL2StrBox.*
Subframe.BushR2StrBox.*

Parameters of the left and right bushing *B7* from the Figure 12.39 between the steering box body and the subframe body, beginning with the specified prefix.

Subframe.BushFL2Vhcl.*
Subframe.BushFR2Vhcl.*

Parameters of the left and right front bushing *B8* from the Figure 12.39 between the vehicle chassis and the subframe body, beginning with the specified prefix.

Subframe.BushRL2Vhcl.*
Subframe.BushRR2Vhcl.*

Parameters of the left and right rear bushing *B9* from the Figure 12.39 between the vehicle chassis and the subframe body, beginning with the specified prefix.

User Accessible Quantities

Please refer to section 24.8.9 'Double wishbone extended suspension' on page 775.

12.7 Additional External Movement

The user has the possibility to apply additional movement of the wheel carrier besides the *Kinematics* and *Compliance*. Typically the additional movement is expressed with translations tx, ty, tz and with rotations rx, ry, rz of the wheel carrier. This movement is in the vehicle fixed frame Fr1.

The external movement can be manipulated either via modification of the corresponding DVA quantities or directly using the parameters of the C code interface *Susp_Ext*, declared in the `include/Car/Susp.h` header file.

Following table shows the user accessible quantities and the belonging parameter in the interface struct:

Table 12.4: External movement quantities and the belonging C-Variable

Quantity	C-Variable	Info
Car.CFL.tx_ext	Susp_Ext.Kin_tExt[0][0]	External translation of the wheel carrier FL in direction x [m]
Car.CFL.ty_ext	Susp_Ext.Kin_tExt[0][1]	External translation of the wheel carrier FL in direction y [m]
Car.CFL.tz_ext	Susp_Ext.Kin_tExt[0][2]	External translation of the wheel carrier FL in direction z [m]
Car.CFL.rx_ext	Susp_Ext.Kin_tExt[0][3]	External rotation of the wheel carrier FL in direction x [rad]
Car.CFL.ry_ext	Susp_Ext.Kin_tExt[0][4]	External rotation of the wheel carrier FL in direction y [rad]
Car.CFL.rz_ext	Susp_Ext.Kin_tExt[0][5]	External rotation of the wheel carrier FL in direction z [rad]
Car.CFR.tx_ext	Susp_Ext.Kin_tExt[1][0]	External translation of the wheel carrier FR in direction x [m]
...	...	...
Car.CRL.tx_ext	Susp_Ext.Kin_tExt[2][0]	External translation of the wheel carrier RL in direction x [m]
...	...	...
Car.CRR.tx_ext	Susp_Ext.Kin_tExt[3][0]	External translation of the wheel carrier RR in direction x [m]
...	...	...

Remark

The velocity of the wheel carrier due to the external movement is not calculated in CarMaker. The user itself can apply additionally besides the translation also the velocity.

Table 12.5: shows the user accessible quantities for the external velocity of the wheel carrier and the corresponding C-Variable in the interface struct:

Table 12.5: External velocity quantities and the belonging C-Variable

Quantity	C-Variable	Info
Car.CFL.txv_ext	Susp_Ext.Kin_vExt[0][0]	External trans. velocity of the wheel carrier FL in direction x [m/s]

Table 12.5: External velocity quantities and the belonging C-Variable

Quantity	C-Variable	Info
...	...	...
Car.CFL.rxv_ext	Susp_Ext.Kin_vExt[0][3]	External rot. velocity of the wheel carrier FL in direction x [rad/s]
...	...	...

Chapter 13

Aerodynamics

13.1 Overview

CarMaker takes into account forces and torques due to external wind loads. All data is conform with the SAE norm J1594.

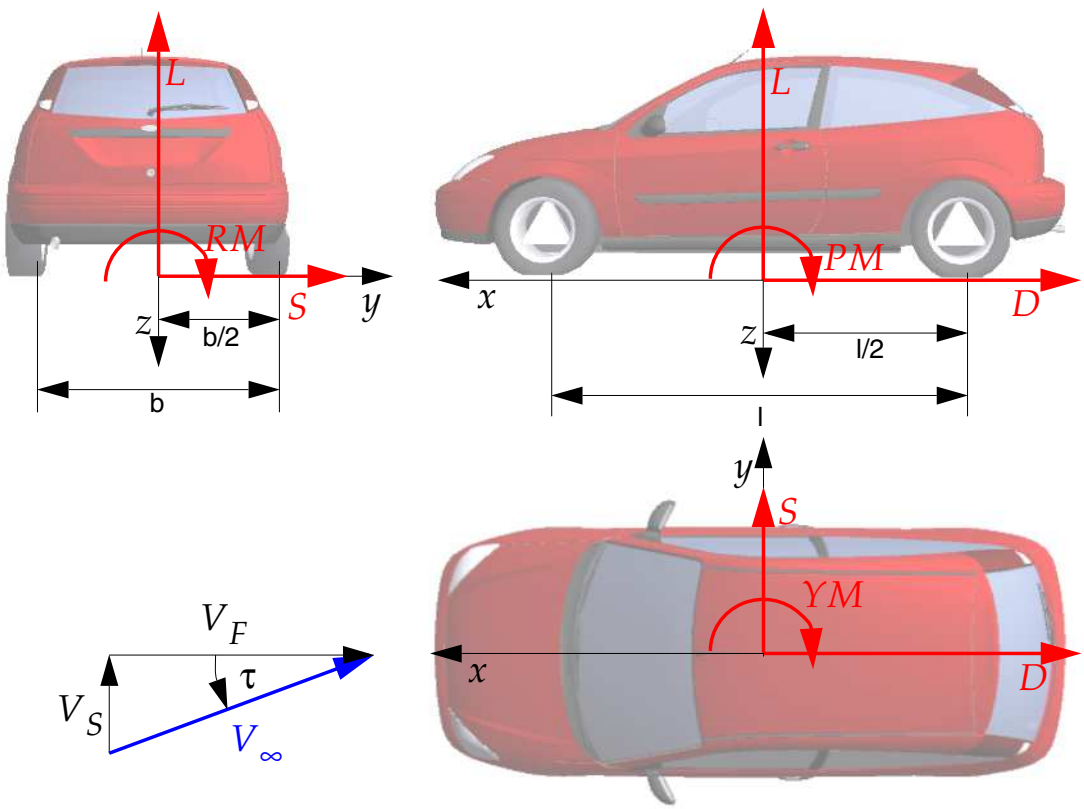


Figure 13.1: CarMaker aerodynamics axis system based on SAE J1594

According to this norm the SAE Road Vehicle Aerodynamics commission defined the following frame (SAE-frame):

- X: positive forward
- Y: positive right
- Z: positive downward
- The origin of the SAE-frame usually in the wheel contact plane at the intersection point of the symmetry lines of track base and wheel base (in design configuration).

Aerodynamic forces and torques depend on:

- Relative wind speed between wind and vehicle (V_∞).
- Angle of attack of wind τ (τ is given in degrees)
  - $\tau = 0$, if the wind is coming from front
  - $\tau > 0$, if the wind is slightly coming from front left.

CarMaker incorporates wind loads as 3 forces and 3 torques on the vehicle body:

$$F_D = -F_x = \frac{\rho}{2} c_D A v_\infty^2 \qquad F_S = F_y = \frac{\rho}{2} c_S A v_\infty^2 \qquad F_L = -F_z = \frac{\rho}{2} c_L A v_\infty^2 \quad \textbf{(EQ 81)}$$

with

F_D	drag force (positive rearward)
c_D	drag coefficient (no dimension)
F_S	side force (positive to right)
c_S	side force coefficient (no dimension)
F_L	lift force (positive upward)
c_L	lift coefficient (no dimension)
ρ	environment air density
A	vehicle reference area
v_∞	relative wind speed

$$M_{RM} = M_x = \frac{\rho}{2} c_{RM} l A v_\infty^2 \qquad M_{PM} = M_y = \frac{\rho}{2} c_{PM} l A v_\infty^2 \qquad M_{YM} = M_z = \frac{\rho}{2} c_{YM} l A v_\infty^2 \quad \textbf{(EQ 82)}$$

with

M_{RM}	rolling moment (positive right side down)
c_{RM}	rolling coefficient (no dimension)
M_{PM}	pitching moment (positive nose up)
c_{PM}	pitching coefficient (no dimension)
M_{YM}	yawing moment (positive nose right)
c_{YM}	yawing coefficient (no dimension)
l	wheel base used as reference length
ρ	environment air density
A	vehicle reference area
v_∞	relative wind speed

13.2 General Parameters

The air density is a global parameter and not specified in this section.

Aero.Marker.pos = ***x*** ***y*** ***z***

This coordinate specified in `FrD` axis system (see section 1.2 'CarMaker Axis Systems') determines the impact of external wind loads. This means when this virtual point reaches the distance of a wind machine the wind takes effect.

Aero.Crosswind.Kind = ***KindStr***

This parameter specifies the kind of crosswind build-up. Possible kinds are:

- *Step:* Crosswind is build-up with a step function if the aerodynamic marker of the vehicle reaches the road marker 'side wind'. The crosswind is completely build-up within one simulation cycle. This kind is default.
- *Linear*: Crosswind is build-up with a linear function if the aerodynamic marker of the vehicle reaches the road marker 'side wind'. The crosswind is completely build-up if the rearmost vehicle point (movie outer skin) is within the road marker area.

Aero.Kind = ***KindStr*** ***VersionId***

This parameter specifies the aerodynamic calculation model.

ModelName	KindStr	Description
1D Look-Up Table	Coeff6x1	aerodynamics model with six 1D Look-Up Tables

Example `Aero.Kind = Coeff6x1 1`

13.3 Models

13.3.1 "1D Look-Up Table"

Aero.pos = ***x*** ***y*** ***z***

Specifies the position of origin of the SAE-frame expressed in the `FrD` axis system (see section 1.2 'CarMaker Axis Systems').

Aero.Ax = ***Area***

The vehicle reference area ***Area*** [m^2] is the projected frontal area including tires and underbody parts.

Aero.Coeff : ***TableWithValues***

this characteristic specifies the 3 force coefficients (EQ 81) and the 3 torque coefficients (EQ 82) depending on the angle of attack of the wind τ (given in degrees!).

The tau-mapping should cover the whole field of angles, ranging from tau = -180 [deg] to tau = +180 [deg]..

Syntax Table mapping with 7 columns

<tau>	c_D	c_S	c_L	c_{RM}	c_{PM}	c_{YM}

Example Aero.mapping:

-180	-0.4	0.0	0.1	0.0	-0.01	0.0
-90	0.0	-1.7	0.9	-0.2	0.0	0.0
0	0.2	0.0	0.1	0.0	-0.03	0.0
90	0.0	1.7	0.9	0.2	0.0	0.0
180	-0.4	0.0	0.1	0.0	-0.01	0.0

Aero.lReference = ***Length***

Optional reference length. Default value is the distance between the aerodynamic marker and the rear point of the movie outer skin.

Chapter 14
Steering System

14.1 Overview

The task of a steer system is to define the driver's "steer influence" to the (front-) suspension. The steering system has an interface that serves two parts. One part is the designated to the driver module (Driver and DrivMan simulate human interactions with the vehicle), the other part interferes to the vehicles suspension module. In case of a driver assistance system which influences the steering system, the Vehicle Control module can be used as interface between driver and steering system.

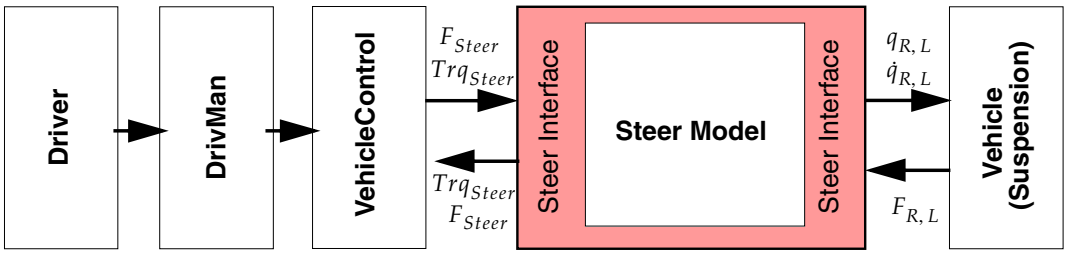


Figure 14.1: Interface of the steering system

Basically steering systems for CarMaker are distinguished by the type of input (control) signals used. The control signal is the output of the driving maneuver module and is an input requirement of the steer system.

Input requirement (Steer by...)	Description
Angle	The steering wheel *angle* is used as input to the steer model. No mass dynamic effects are regarded. This means that the update to new values of steer angles happens infinitely fast (no differential equation).

Input requirement (Steer by...)	Description
Torque	Input is the steering torque from the driving maneuver module. A differential equation has to be used to *calculate* the steering wheel angle. Mass dynamic effects should be regarded by the model. Infinitively fast changes of steering wheel angles are not possible.

14.1.1 Steering Interface

There are two interface structs defined for the information exchange between the steering model and the simulation program, one for the initialization and one for the evaluation function.

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tSteeringCfgIF*:

Input Variable	Unit	Description
Ang	rad	Initial steering wheel angle
PosSign		Sign of rack translation / steering knuckle rotation [-1,+1] (depends on front susp. kinematics)

Output Variable	Unit	Description
SInputKind		Supported input kind by the steering model: 0: steering by angle 1: steering by torque 2: steering by angle and torque
PrefByDriver		Preferred input kind by the driver: 0: steering by angle,1: steering by torque

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tSteeringIF*:

Input Variable	Unit	Description
essential inputs		
SteerBy		Steering mode: by angle or by torque
Ang AngVel AngAcc	rad rad/s rad/s^2	Steering wheel angle Steering wheel rotational velocity Steering wheel rotational acceleration
Trq	Nm	Steering wheel torque
L.Frc R.Frc	N	Suspension forces to tie rod translation (left and right)
L.Inert R.Inert	kg	Suspension inertias to tie rod translation (left and right)
additional inputs		
AssistFrc_Ext	N	External assistance force at rack
AssistTrqCol_Ext AssistTrqPin_Ext	Nm	External assistance torque at column/pinion

Output Variable	Unit	Description
essential outputs		
L.q, R.q L.qp, R.qp L.qpp, R.qpp	m m/s m/s^2	Steering rack translation Steering rack trans. velocity Steering rack trans. acceleration
L.iSteer2q R.iSteer2q		Ratio steer rotation to rack translation left/ right
Ang AngVel AngAcc	rad rad/s rad/s^2	Steering wheel angle Steering wheel angle velocity Steering wheel angle acceleration
additional outputs		
Trq	Nm	Steering wheel torque
TrqStatic	Nm	Steering wheel torque required for static conditions (no acceleration)
AssistFrc	N	Assistance force at rack
AssistTrqCol AssistTrqPin	Nm	Assistance torque at column/pinion

The variables for steering wheel angle and torque are both, inputs and outputs. Depending on current steering mode (steer by angle or steer by torque), these are used as inputs or outputs. In the next chapters the both modes are explained in detail.

14.1.2 Steer by Angle

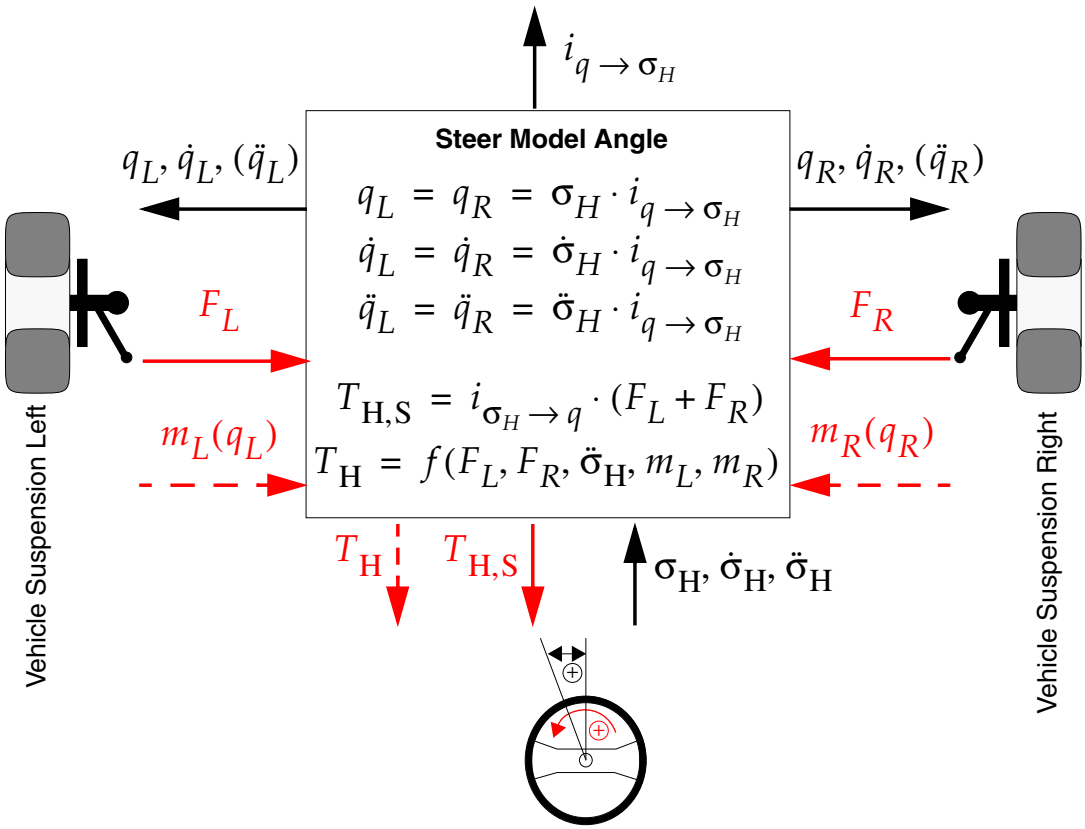


Figure 14.2: Principle of Steer by Angle Models

Figure 14.2 shows the basic functionality of a steer by angle model. The steering wheel angle σ_H and its derivatives are inputs to the steer model. The steering wheel angle is translated to the generalized quantity $q = q_L = q_R$. The same way for the derivatives. The translation is defined as follows:

$$i_{q \to \sigma_H}(\sigma_H) = \frac{1}{i_{\sigma_H \to q}(\sigma_H)} = \frac{\dot{q}}{\dot{\sigma}_H} \qquad \text{(EQ 83)}$$

(EQ 83) clarifies that the translation might depend on the steering wheel angle. Such a variable steer translation is commonly used with modern steer systems.

Optionally, the static steering wheel torque $T_{H,S}$ can be returned by the model depending on ratio and suspension forces to the tie road: $T_{H,S} = i_{\sigma_H \to q} \cdot (F_L + F_R)$. For example, this torque could be used for torque feedback.

Additionally, if a differential equation exists, the dynamic steering wheel torque T_H can be returned depending on steering wheel rotation (angle, velocity, acceleration) and suspension forces / inertias to tie rod: $T_H = f(F_L, F_R, \ddot{\sigma}_H, m_L, m_R)$.

14.1.3 Steer by Torque

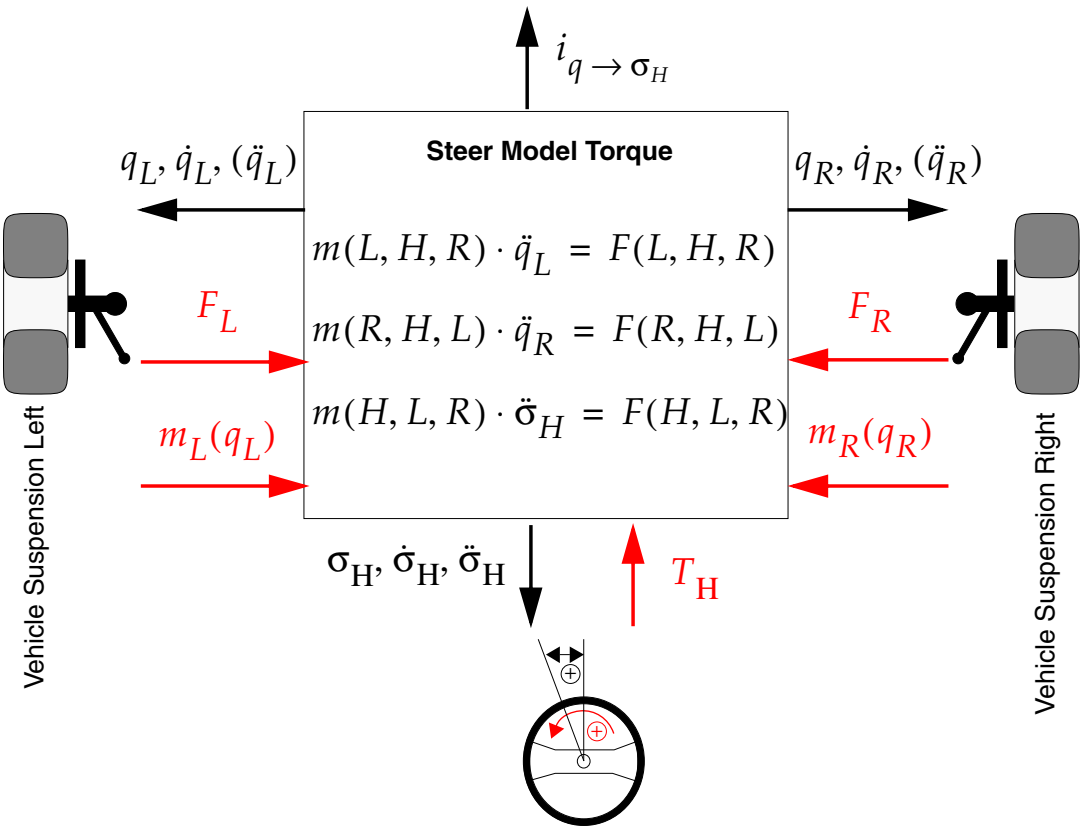


Figure 14.3: Principle of Steer by Torque Models

A steer by torque model is characterized by the input of the steering wheel torque to the steer model and that the steer angle and velocity and acceleration is returned by the steer model. In this case differential equations as depicted in Figure 14.3 have to be calculated.

If a stiff rack steering system is modeled, the generalized steer coordinates reduce to one $q_{\text{Steer}} = q_L = q_R$. Then the differential equations can be written as

$$(m_L + m_R + m_H) \cdot \ddot{q}_{\text{Steer}} = F_L + F_R + F_H . \qquad \textbf{(EQ 84)}$$

The acceleration of the steering wheel is determined from

$$\ddot{\sigma}_H = i_{q \to \sigma_H} \cdot q_{\text{Steer}} . \qquad \textbf{(EQ 85)}$$

This simple approach of steer by torque model can be extended by detailed models with steering boosters or with extra functionality like active steering mechanisms.

14.1.4 Steering-in-the-Loop test bench

If using as steering model a real steering system in a test bench, the input (desired or target) signals like steering wheel angle, steering wheel torque or suspension forces to tie rod or rack differ from the current state variables. These variables are controlled to reach the input values, but there is a build-up time delay.

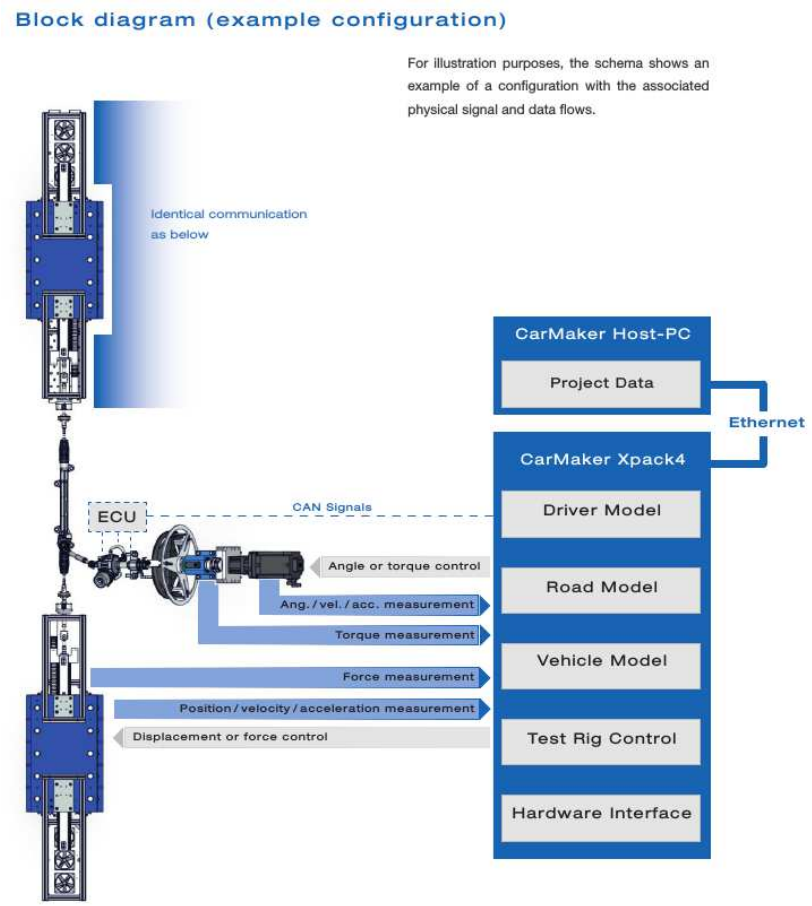


Figure 14.5: Steering in-the-loop test bench

For visualization of both, the input and the current values, the user example steering model `<Project>/src/MySteering.c` was extended with further signals and corresponding quantities. Below the available quantities are illustrated:

Name UAQ	Unit	Description
Steer.In.L.Frc Steer.In.R.Frc	N	Input (desired) force from suspension to tie rod (left, right side)
Steer.L.Frc Steer.R.Frc	N	Current force from suspension to tie rod (left, right side)
Steer.In.Rack.Frc	N	Input (desired) total suspension force to rack
Steer.Rack.Frc	N	Current total suspension force to rack
Steer.In.WhlAcc	rad/s^2	Input (desired) steering wheel rotational acceleration
Steer.In.WhlAng	rad	Input (desired) steering wheel angle
Steer.In.WhlTrq	Nm	Input (desired) steering wheel torque
Steer.In.WhlVel	rad/s	Input (desired) steering wheel rotational velocity

14.2 General Steering System Parameters

Steering.Kind = ***KindStr*** ***VersionId***

KindStr specifies the steering model to be used. The CarMaker steering model library provides the following models:

ModelName	KindStr	Description
Static Steer Ratio	GenAngle	Standard steer by angle model with 0 DOF.
Dynamic Steer Ratio	GenTorque	Standard steer by torque model with 1 DOF.
Pfeffer with Power Steering	Pfeffer	Steering model of Dr. Pfeffer, University of Munich, with 2 DOF and power assistance module.

Example
```
Steering.Kind = GenAngle 1
```

Steering.Ratio.Kind = ***RatioKind***

Specifies the kind of steering gear ratio. Possible options are:

- `Constant`: constant steering gear ratio; default option.
- `Variable`: non-linear steering gear ratio defined with a data table.
- `Mapping`: non-linear steering pinion angle-rack travel characteristic.

Steering.Rack2StWhl = ***value***
Steering.Rack2StWhl : ***TableRatio [PinionAng Ratio]***
Steering.Rack2StWhl : ***TableRack [PinionAng RackTravel]***

Specifies the steering gear ratio. This parameter can be a constant value for the relationship: Steering Gear Ratio: PinionAngle [rad] = Rack2StWhl [rad/m] * RackTranslation [m] or a two-dimensional data table characteristic:

Example
```
Steering.Rack2StWhl = 100
Steering.Rack2StWhl:
   -60.0 110
     0.0 80
   +60.0 110
Steering.Rack2StWhl:
   -60.0 0.044
     0.0 0.0
   +60.0 -0.044
```

It's recommended to specify a ratio or rack travel for pinion angle 0.0 and not to exceed 50 as characteristic poins number.

Steering.SteerByTorque = ***bool***

Optional. Usually each CarMaker steering model sets the steering kind to be used by the IPGDriver. With this parameter the steering kind can be overwritten or set (e.g: for simulink).

14.3 Steering System “Static Steer Ratio”

This steering system of CarMaker is of the kind steer by angle (see section 14.1.2).

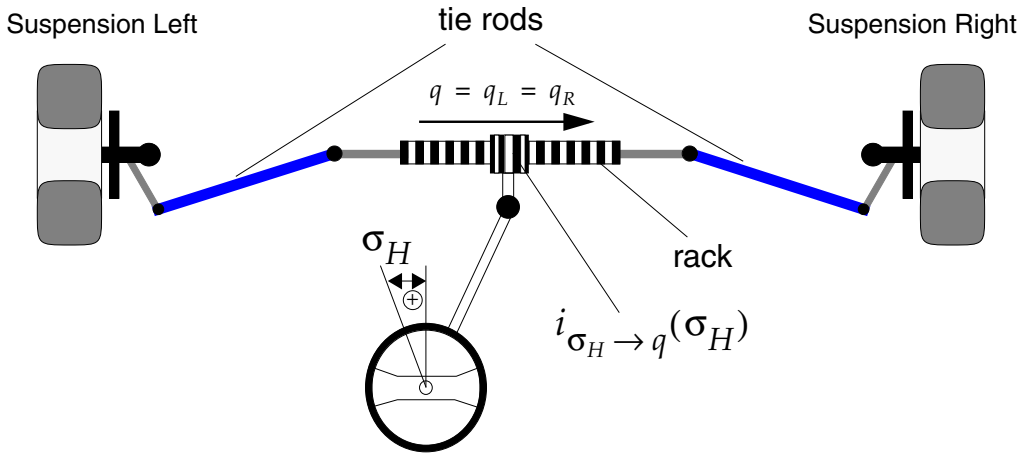


Figure 14.7: Steer system ‘Static Steer Ratio’

This model simulates a simple rack steering system. The whole steering system is assumed to be stiff. The generalized coordinates on the left and the right side are equal $q_{\text{Steer}} = q_L = q_R$.

The rack position q_{Steer} is calculated with the steering gear ratio:

$$q_{\text{Steer}} = \frac{\sigma_H}{i_{\sigma_H \to q}(\sigma_H)} \qquad \text{(EQ 86)}$$

Using a steering ratio characteristic, the rack position q_{Steer} is calculated while vehicle initialisation. For this, the gear ratio within the values is supposed to be linear:

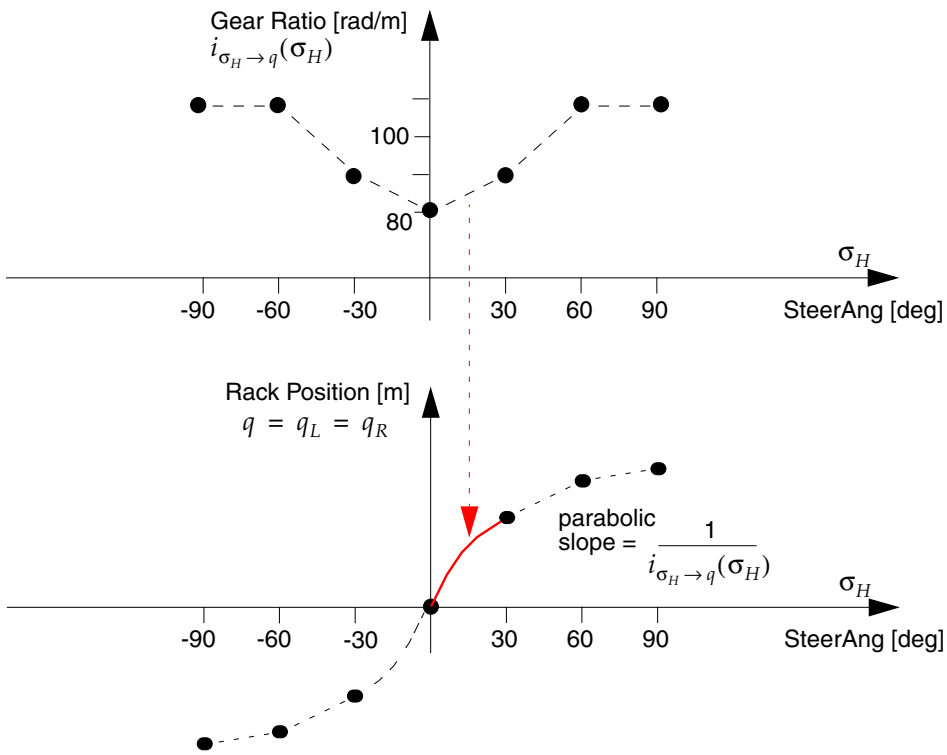


Figure 14.8: Rack position with steering ratio characteristic

A more detailed model of the rack position is possible with more values.

14.4 Steering System “Dynamic Steer Ratio”

This steering model is a model of the kind steer by torque (see section 14.1.3).

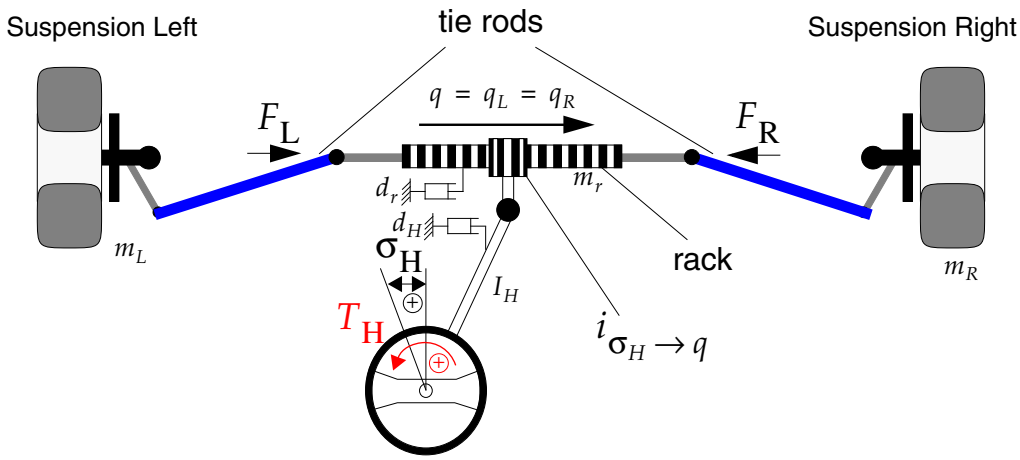


Figure 14.9: Steer system ‘Dynamic Steer Ratio’

This model simulates a simple steering system with one degree of freedom q_{Steer}. The whole steering system is assumed to be stiff. The generalized coordinates on the left and the right side are equal $q_{\text{Steer}} = q_L = q_R$.

This model calculates the generalized coordinate q_{Steer} with the following relation:

$$(m_L + m_R + m_H) \cdot \ddot{q}_{\text{Steer}} = F_L + F_R + F_H + F_{assist} - d_r \cdot \dot{q}_{Steer} \quad \textbf{(EQ 87)}$$

with the steering mass

$$m_H = I_H \cdot i^2_{\sigma_H \to q} + m_r \quad \textbf{(EQ 88)}$$

and with the steering force

$$F_H = (T_H + M_{assistC} + M_{assistP} - d_H \cdot \dot{\sigma}_H) \cdot i_{\sigma_H \to q}. \quad \textbf{(EQ 89)}$$

The assisting torque at column is calculated with the driver torque T_H and an amplification factor: $M_{assistC} = T_H \cdot (Amp - 1)$. The assistance force at rack F_{assist} and the assistance torque at pinion $M_{assistP}$ can be additionally applied by an external model.

The steering angle σ_H is calculated with the steering gear ratio:

$$\sigma_H = q_{Steer} \cdot i_{\sigma_H \to q} \quad \textbf{(EQ 90)}$$

Steering.Amplify = ***value***

Specifies the amplification of the driver steering torque. Corresponds to the parameter Amp in (EQ 89). Unit: [-]. Default: 3.0.

Steering.Wheel.I = ***value***

Steering inertia of all rotating parts (wheel, column, ...). Corresponds to the parameter I_H in (EQ 88). Unit: [kgm^2]. Default: 0.001.

Steering.Wheel.d = ***value***

Damping coefficient of rotating part (wheel, column, ...). Corresponds to the parameter d_H in (EQ 89). Unit: [Nms/rad]. Default: 1.0.

Steering.Rack.mass= ***value***

Steering rack mass. Corresponds to the parameter m_r in (EQ 88).
Unit: [kg]. Default: 10.0.

Steering.Rack.d = ***value***

Damping coefficient of steering rack. Corresponds to the parameter d_r in (EQ 87).
Unit: [Ns/m]. Default: 50.0.

Steering.Rack.travel = ***value***

Specifies the movement range (min-max travel) of the steering rack.
Unit: [m]. Default: 0.08.

14.5 Steering System “Pfeffer”

14.5.1 Overview

The *Pfeffer* steering model consists of a mechanical module and an assisting module (see Figure 14.10). The mechanical module models the mechanical parts of a steering system including steering wheel, steering column, torsion bar, hardy disc, mesh, steering rack and pinion. The assisting module refers to the hydraulic powered steering (HPS) and the electrical powered steering (EPS) system.

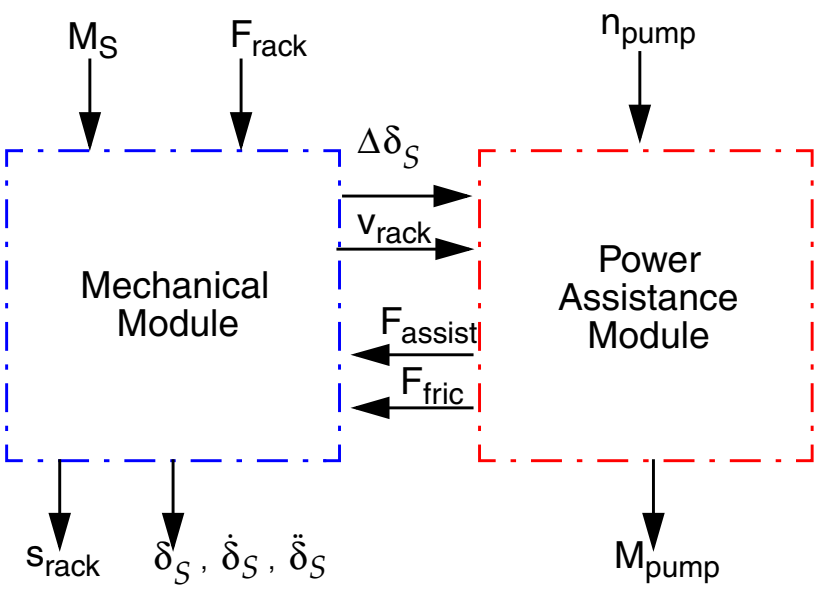


Figure 14.10: Steering model layout *SteerByTorque* with HPS assistance (δ_S - steering wheel angle; F_{rack} - rack force; v_{rack}- steering rack velocity; $\Delta\delta_S$ - torsion bar twist angle; F_{assist} - assistant force of the power assistance module; F_{fric} - frictional force of the hydraulic seals; $\dot{\delta}_S$ - steering wheel velocity; $\ddot{\delta}_S$ - steering wheel acceleration; s_{rack}- steering rack displacement; M_S - steering wheel torque; M_{pump} - pump torque, n_{pump} - pump rotational velocity)

14.5.2 The Mechanical Module

The mechanical steering model is shown in Figure 14.11 schematically. It contains all mechanical components which transfer torque from the steering wheel to the tie rods, i.e. the steering wheel, steering column, hardy disc, torsion bar, pinion, steering rack and the occurring damping and friction effects.

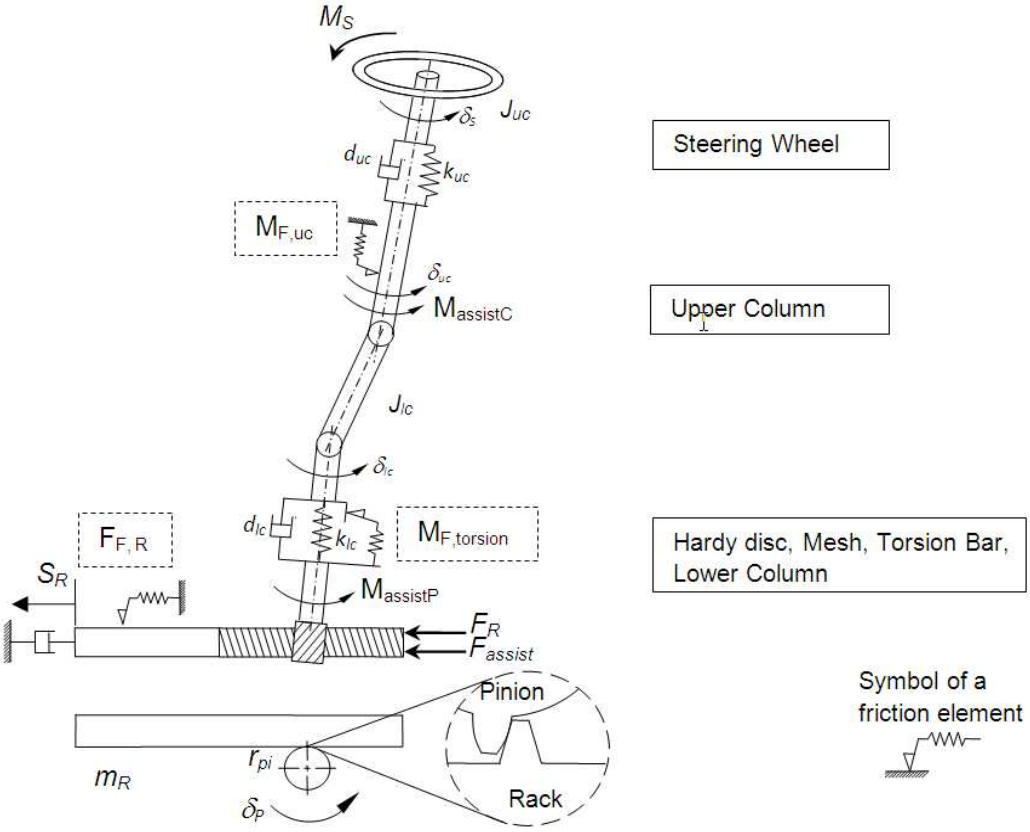


Figure 14.11: Mechanical Module of the Steering Model for all power assistance modules except EPSc; uc: upper column; lc: lower column; S: steering wheel; R: steering rack; T: torsion bar

The equations representing the three degrees of freedom (steering wheel rotation δ_S, upper column rotation δ_{uc} and rack position s_R) of the steering system are listed below:

$$J_{uc}\ddot{\delta}_S + M_{uc} + M_{damp} = M_S \qquad \textbf{(EQ 91)}$$

$$J_{lc}\ddot{\delta}_{uc} + M_{F,uc} + (M_{lc} + M_{F,T})\frac{\dot{\delta}_{lc}}{\dot{\delta}_{uc}} = M_{uc} + M_{assistC} \text{ with } \delta_{lc} = f(\delta_{uc}) \qquad \textbf{(EQ 92)}$$

$$m_R\ddot{s}_R + F_{F,R} + F_{damp} = \frac{M_{assistP} + M_{F,T} + M_{lc}}{r_{pin}} + F_{assist} + F_R \qquad \textbf{(EQ 93)}$$

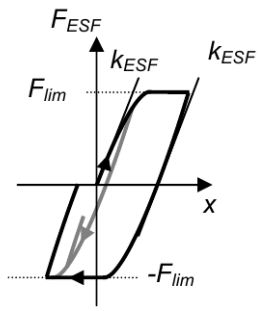
$$M_{uc} = k_{uc}(\delta_S - \delta_{uc}) + d_{uc}(\dot{\delta}_S - \dot{\delta}_{uc}),\ M_{lc} = k_{lc}(\delta_{lc} - \delta_p) + d_{lc}(\dot{\delta}_{lc} - \dot{\delta}_p) \qquad \textbf{(EQ 94)}$$

M_{lc} is the torque of the spring damper element, which considers the flexibility of the lower shaft components (torsion bar, mesh, hardy disc and lower column). M_{uc} is the torque of the spring damper element, which considers the flexibility of the upper column.

F_{damp} is the steering rack damping force and M_{damp} is the column damping torque. F_R is the steering rack force calculated by the CarMaker vehicle model. F_{assist} is the assisting force supplied by the power assistance module HPS or EPSapa. $M_{assistC}$ is the assisting torque at column and $M_{assistP}$ is the assisting torque at pinion supplied by the assistance module EPSc.

Depending on selected power assistance module one of the three assistance values is calculated. In the case if the assistance torque at column is applied using EPSc module, the torsion bar is located at the upper end of the steering shaft and $M_{assistC}$ applies below the torsion bar.

M_F and F_F are the lumped torque and force friction, calculated with the Exponential-Spring-Friction-Element (ESF-Element):



$$F_{ESF} = F_{lim}(1 - e^{-(f_{ESF} \cdot x)}) \quad \text{(EQ 95)}$$

$$\text{with } f_{ESF} = \frac{k_{ESF}}{F_{lim}} \quad \text{(EQ 96)}$$

The ESF-Element requires the stiffness at zero force k_{ESF} and the force limit F_{lim} as input parameters. For details please refer to the publication *Modelling of a hydraulic steering system, (Pfeffer, Harrer, Johnston, 2006).*

Column non-uniformity

The lower column angle δ_{lc} is related to the upper column angle δ_{uc} by a function $f(\delta_{uc})$ describing the column non-uniformity due to the double universal joints. The relation can be described by a non-linear map or by defining the upper and lower column mount positions shown in the Figure 14.12:

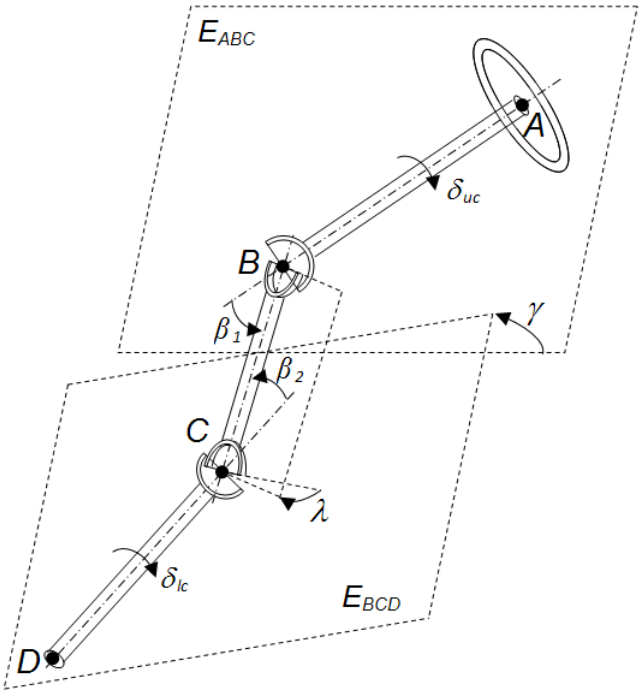


Figure 14.12: Steering shaft with two universal joints

With the positions A, B, C and D the deflection angles β_1 and β_2 are calculated as follows:

$$\cos(\beta_1) = \frac{\overrightarrow{AB} \cdot \overrightarrow{BC}}{\|\overrightarrow{AB}\| \cdot \|\overrightarrow{BC}\|}, \cos(\beta_2) = \frac{\overrightarrow{BC} \cdot \overrightarrow{CD}}{\|\overrightarrow{BC}\| \cdot \|\overrightarrow{CD}\|} \quad \text{(EQ 97)}$$

Additionally the fork angle λ is given for the difference angle between the two fork planes of the intermediate axis.

The frame planes E_{ABC} and E_{BCD}, which are given by the three corresponding points, are twisted with the angle γ:

$$\cos(\gamma) = \frac{\vec{n}_{ABC} \cdot \vec{n}_{BCD}}{\|\vec{n}_{ABC}\| \cdot \|\vec{n}_{BCD}\|} \text{ with } \vec{n}_{ABC} = \overrightarrow{AB} \times \overrightarrow{BC}, \vec{n}_{BCD} = \overrightarrow{BC} \times \overrightarrow{CD} \quad \text{(EQ 98)}$$

Referred to Florea Duditza (VDI, ’Kardangelenkgetriebe und ihre Anwendungen’) the lower column angle is calculated as follows:

$$\delta_{lc} = \operatorname{atan}\left(\frac{c_1 \tan(\delta_{uc})}{c_2 + c_3 \tan(\delta_{uc})}\right) \text{ with } \begin{aligned} c_1 &= \cos(\beta_2)(1 + \tan(\lambda - \gamma)^2) \\ c_2 &= \cos(\beta_1)(1 + \cos(\beta_2)^2 \tan(\lambda - \gamma)^2) \\ c_3 &= (\cos(\beta_2)^2 - 1)\tan(\lambda - \gamma) \end{aligned} \quad \text{(EQ 99)}$$

Steer by Angle and Steer by Torque

The mechanical module of the *Pfeffer* steering model supports as input from the driving maneuver module (DrivMan) the steering wheel torque M_S for steering by torque or the steering wheel angle δ_S (and accordingly $\dot{\delta}_S$, $\ddot{\delta}_S$) for steering by angle. In the second case the steering wheel torque M_S is only an internal information and isn’t required by the driving maneuver module.

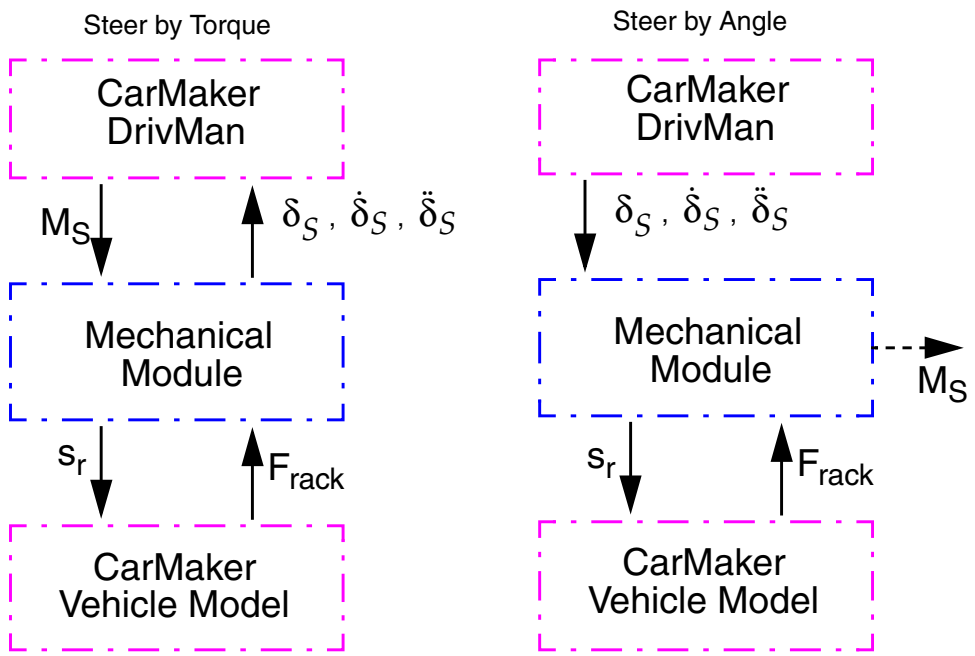


Figure 14.13: Steer by Torque and Steer by Angle

14.5.3 The Power Assistance Module

The power assistance module is a model which represents the assistance for the mechanical module of a steering system to reduce the steering effort from driver. There are three main types - Hydraulic (HPS) and Electrical (EPSc and EPSapa) systems. A HPS system uses hydraulic pressure supplied by an engine-driven pump to assist the motion of turning the steering wheel. An EPSapa system uses an electric motor coupled directly to the steering rack for assistance. An EPSc system uses an electric motor coupled directly to the steering column shaft for assistance.

In addition using the DVA mechanism the user can implement his own assistance model (e.g. EHPS) or steer without assistance. Furthermore an external assistance force/torque offset can be applied to each power assistance model using the corresponding external assistance quantities.

HPS (hydraulic powered system)

The hydraulic pressure is determined by the boost curve of the power assisting system which is implemented by a look-up table (Figure 14.14). Together with the torsion bar torque (corresponds roughly to the steering wheel torque) the steady state pressure difference of the left and right cylinder chamber can be determined.

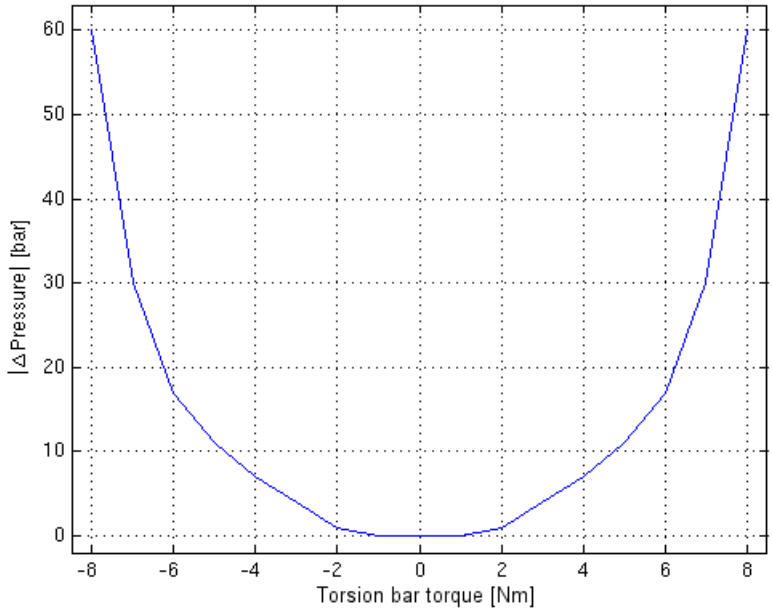


Figure 14.14: Hydraulic boost curve

The readout steady state pressure can be applied without time lag or with a first order time lag in order to approximate the hydraulic system dynamics.

Figure 14.15 shows the underlying model assumption about hydraulic system dynamics.

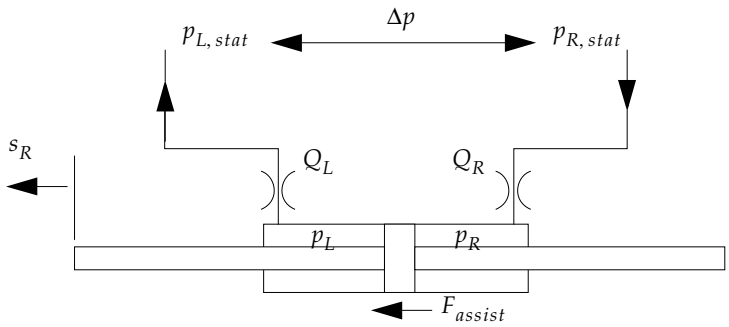


Figure 14.15: Low order hydraulic model

The pressure of the pressurized side is equal to Δp out of the boost curve, on the other side the pressure is assumed to be zero. The pressure difference between the static external pressure and the internal chamber pressure is used to work out the flow with the turbulent orifice equation. Written down for the right hand side it is:

$$p_{R,\,stat} - p_R = \frac{\varsigma}{2} \cdot \left(\frac{Q_R}{C_q \cdot A_O} \right)^2 \qquad \textbf{(EQ 100)}$$

And with the fluid compressibility equation the flow can be calculated as

$$\frac{V_R}{\beta_F} \cdot \frac{dp_R}{dt} = Q_R - Q_{out} = Q_R - \dot{s}_R \cdot A_P \qquad \textbf{(EQ 101)}$$

Combining both equations leads to

$$\frac{V_R}{\beta_F} \cdot \frac{dp_R}{dt} = C_q \cdot A_O \cdot \sqrt{\frac{2}{\varsigma}(p_{R,stat} - p_R)} - \dot{s}_R \cdot A_P \quad \text{(EQ 102)}$$

This equation is used to model the time delayed pressure build-up in both chambers. For details please refer to the publication *Modelling of a hydraulic steering system, (Pfeffer, Harrer, Johnston, 2006).*

The assisting force is calculated with the pressure difference:

$$F_{assist} = (p_R - p_L) \cdot A_P \quad \text{(EQ 103)}$$

EPSapa (axle parallel drive)

An electrical powered steering system ’Axle Parallel Drive EPS’ is modelled for the EPSapa assistance module. The assist power is applied directly to the rack with a belt and ball nut mechanism. The electric powered assistance module can be modelled in either a simplified or an extensive model. The simplified model is a low order model consisting of a look-up table while the transmission between the motor and the steering rack is assumed to be without compliance. The extensive model is a high order model including the compliances along the transmission line, friction effects and a controller for the motor current.

EPSapa simplified (EPSapa_s)

The assisting torque of the EPSapa is determined by the boost curve according to the torsion bar torque. The torsion bar torque is the measured twist torque in the torsion bar due to the torsion bar twist angle. Firstly the current of the motor i_M can be determined from the characteristic curve (see Figure 14.16).

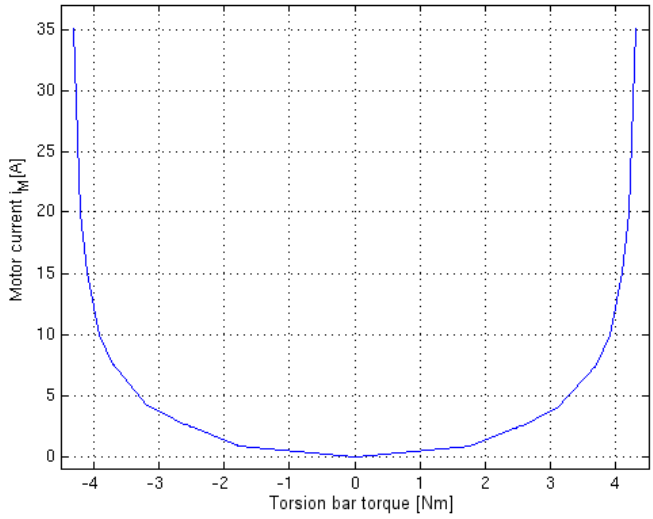


Figure 14.16: Electrical boost curve

The effect of vehicle speed on the system is considered by using a three-dimensional boost mapping depending on vehicle velocity and torsion bar torque.

Then the assisting force is calculated by the ratio of the axle parallel drive:

$$M_E = i_M \cdot K_t \quad \text{(EQ 104)}$$

$$F_{assist} = (M_E + M_{Eext}) \cdot \frac{I_{belt}}{I_{KGT}} \quad \text{(EQ 105)}$$

EPSapa extensive (EPSapa_e)

The high order electrical powered model is shown in Figure 14.17. The belt and the linkage between ball nut mechanism and the steering rack are represented by the spring-damper model.

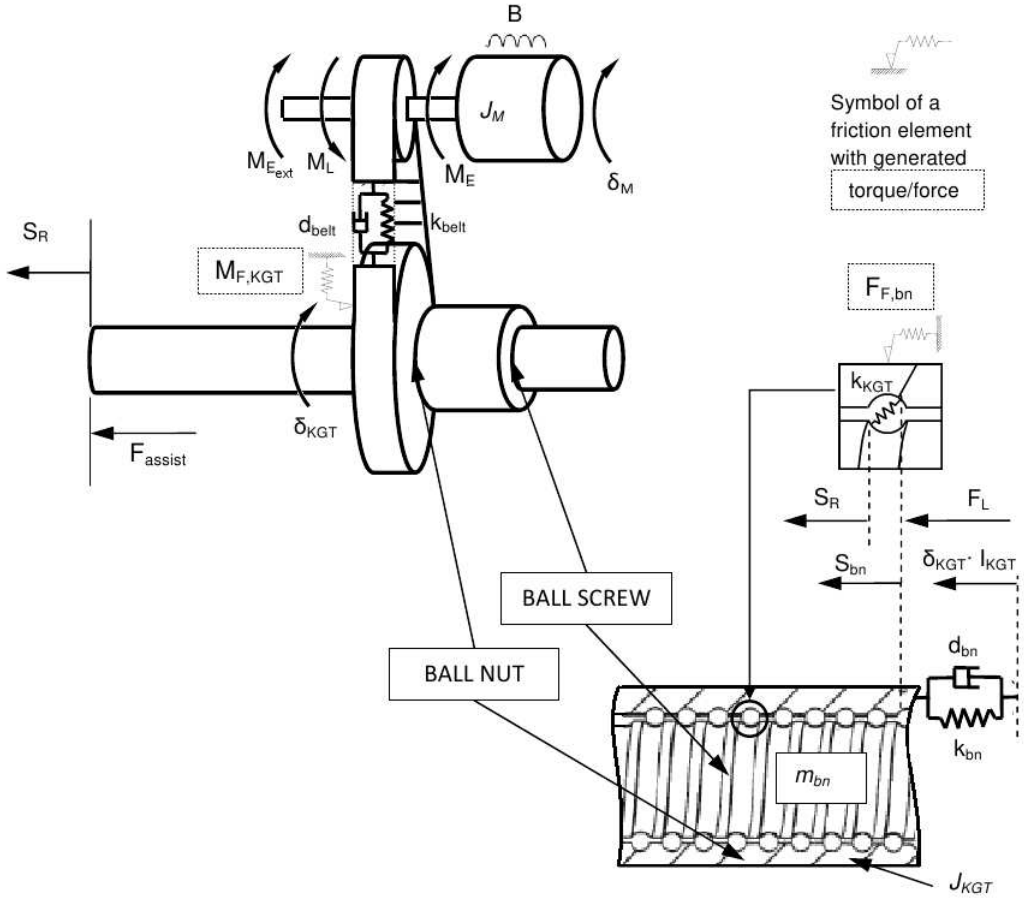


Figure 14.17: High order electrical powered model

The equations used are listed as following:

$$M_L = k_{belt}(\delta_M - \delta_{KGT} \cdot I_{belt}) + d_{belt}(\dot{\delta}_M - \dot{\delta}_{KGT} \cdot I_{belt}) \quad \textbf{(EQ 106)}$$

$$J_M \cdot \ddot{\delta}_M = M_E + M_{Eext} - M_L - \dot{\delta}_M \cdot B \quad \textbf{(EQ 107)}$$

$$F_L = k_{bn}(\delta_{KGT} \cdot I_{KGT} - s_{bn}) + d_{bn}(\dot{\delta}_{KGT} \cdot I_{KGT} - \dot{s}_{bn}) \quad \textbf{(EQ 108)}$$

$$J_{KGT} \cdot \ddot{\delta}_{KGT} = M_L \cdot I_{belt} - M_{F,KGT} - F_L \cdot I_{KGT} \quad \textbf{(EQ 109)}$$

$$F_{assist} = k_{KGT}(s_{bn} - s_R) + d_{KGT}(\dot{s}_{bn} - \dot{s}_R) \quad \textbf{(EQ 110)}$$

$$m_{bn} \cdot \ddot{s}_{bn} = F_L - F_{F,bn} - F_{assist} \quad \textbf{(EQ 111)}$$

$M_{F,KGT}$ and $F_{F,bn}$ are calculated by an ESF friction model.

The EPSapa model uses a PI-Controller to achieve the demanded torque M_{demand} derived from boost curve by tuning the motor current i_M (see Figure 14.18).

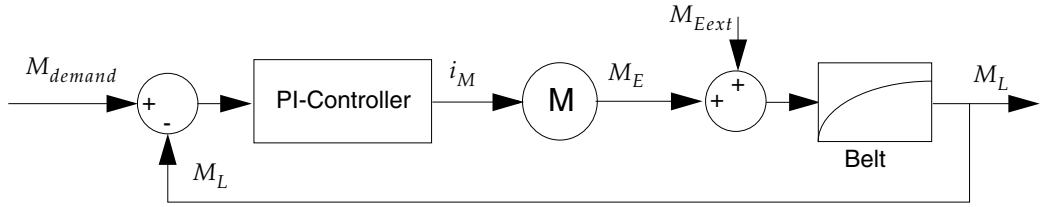


Figure 14.18: The EPSapa steering controller

The summarized motor propulsion moment M_E and an external torque M_{Eext} drive the motor shaft, whose motion generates a spring-damper moment M_L in the belt. The additional torque M_{Eext} non zero would provoke a decline of the motor current i_M.

EPSc (column or pinion)

The electrical powered steering system EPSc is used to calculate the assist torque at the steering column or pinion.

For the EPSc power assistance module with assist torque at column the torsion bar is included in the upper part:

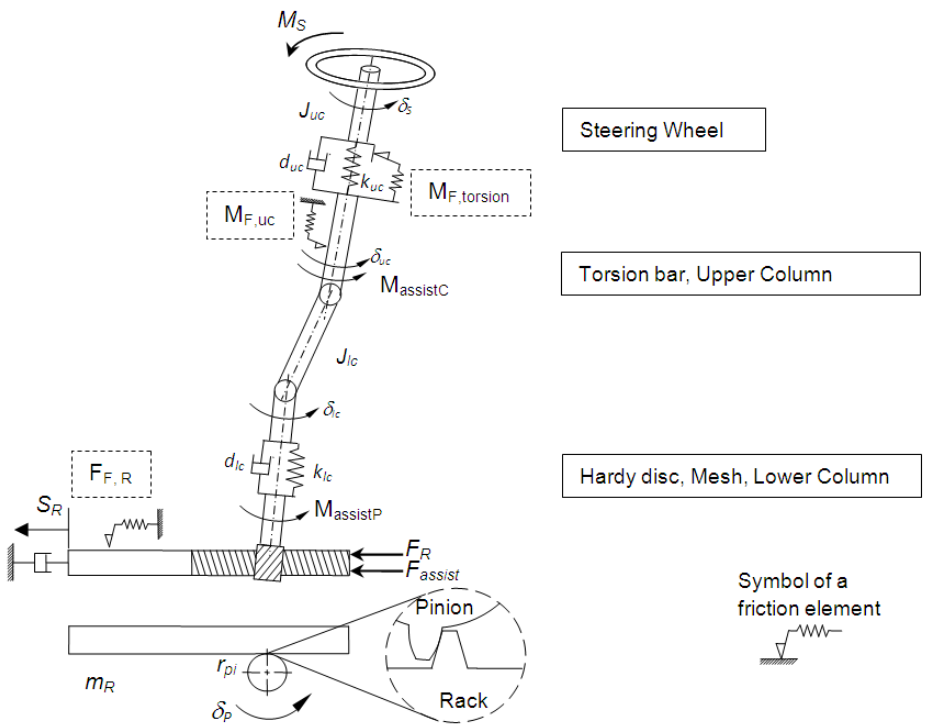


Figure 14.19: Mechanical Module of the Steering Model with EPSc power assistance module; uc: upper column; lc: lower column; S: steering wheel; R: steering rack; T: torsion bar; P: pinion

The position of the friction torque $M_{F,T}$ changes in the following equations:

$$J_{uc}\ddot{\delta}_S + M_{uc} + M_{F,T} + M_{damp} = M_S \qquad \text{(EQ 112)}$$

$$J_{lc}\ddot{\delta}_{uc} + M_{F,uc} + M_{lc}\frac{\dot{\delta}_{lc}}{\dot{\delta}_{uc}} = M_{uc} + M_{F,T} + M_{assistC} \qquad \text{(EQ 113)}$$

$$m_R\ddot{s}_R + F_{F,R} + F_{damp} = \frac{M_{assistP} + M_{lc}}{r_{pin}} + F_{assist} + F_R \qquad \text{(EQ 114)}$$

Like the EPSapa model it can be modelled as either a simplified model or an extensive model. The simplified model is a low order model consisting of a look-up table while the transmission between the motor and the column (pinion) is assumed to be without compliances. The extensive model includes an additional degree of freedom for the motor rotation and a controller for the motor current.

EPSc simplified (EPSc_s)

The assisting torque of the EPSc is determined in the same manner like for EPSapa by the boost curve of the motor current according to the torsion bar torque (see Figure 14.16):

$$M_E = i_M \cdot K_t \quad \text{(EQ 115)}$$

$$M_{assist} = (M_E + M_{Eext}) \cdot I_{belt} \quad \text{(EQ 116)}$$

Depending on the position of the assisting torque (column or pinion) the following equations would be changed. For the assisting torque at pinion $M_{assistP} = M_{assist}$ and for the assisting torque at column $M_{assistC} = M_{assist}$.

EPSc extensive (EPSc_e)

The extensive EPSc model includes like the EPSapa model an degree of freedom for the motor rotation. The assist torque is transmitted from the motor by worm gear or belt drive. The compliance of this mechanism is modelled with a spring-damper model with stiffness k_{belt} and damping coefficient d_{belt}. The equations for the EPSc would be:

$$J_M \cdot \ddot{\delta}_M = M_E + M_{Eext} - M_L - \dot{\delta}_M \cdot B \quad \text{(EQ 117)}$$

Depending on the position of the assisting torque (column or pinion) the following equations would be changed. For the assisting torque at pinion:

$$M_L = k_{belt}(\delta_M - s_R I_{belt}/r_{pin}) + d_{belt}(\dot{\delta}_M - \dot{s}_R I_{belt}/r_{pin}) \quad \text{(EQ 118)}$$

$$M_{assistP} = M_L \cdot I_{belt} \quad \text{(EQ 119)}$$

For the assisting torque at column:

$$M_L = k_{belt}(\delta_M - \delta_{uc}) + d_{belt}(\dot{\delta}_M - \dot{\delta}_{uc}) \quad \text{(EQ 120)}$$

$$M_{assistC} = M_L \cdot I_{belt} \quad \text{(EQ 121)}$$

The EPSc model uses also a PI-Controller to control the desired torque from boost curve (see Figure 14.18).

14.5.4 Steering System “Pfeffer” Parameters

Steering.OSRate = *value*

Specifies the integration substeps in one simulation cycle of the whole steering model. The value is referred to 1ms cycle time. If an other cycle time is specified, the integration substep is adapted internally to the specified cycle time.
Default: 5. Range: 1-200. For the extensive power assistance models 25 as minimum value is taken automatically.

Parameters for Mechanical Module

The parameters for the steering gear ratio are described in section 14.2 ’General Steering System Parameters’.

Steering.Mech.Column.I = *value*

Specifies the total inertia of upper column parts. Corresponds to the parameter J_{uc} in (EQ 91). For the power assistance module EPS at column it contains the inertia of steering wheel, upper column and torsion bar. For all other power assistance modules it contains the inertia of steering wheel and upper column.
Unit: [kgm^2]. Default: 0.026.

Steering.Mech.Column.I_im = *value*

Specifies the total inertia of lower column parts. Corresponds to the parameter J_{lc} in (EQ 92). For the power assistance module EPS at column it contains the inertia of hardy disc and intermediate shaft. For all other power assistance modules it contains the inertia of hardy disc, intermediate shaft and torsion bar.
Unit: [kgm^2]. Default: 0.001.

Steering.Mech.Column.k = *value*

Specifies the stiffness of the steering column. For the power assistance module EPS at column this stiffness is combined with the stiffness of other parts to the stiffness k_{lc} in (EQ 94). For all other power assistance modules this stiffness is k_{uc}. Unit: [Nm/deg]. Default: 12.

Steering.Mech.Column.Fric.k = *value*

Specifies the stiffness of the column exponential spring friction model. Corresponds to the parameter k_{ESF} in (EQ 96). Unit: [Nm/rad]. Default: 7000.

Steering.Mech.Column.Fric.M_max = *lowerlimit upperlimit*

Specifies the upper and lower limit of the column exponential spring friction torque. Corresponds to the parameter F_{lim} in (EQ 95) and (EQ 96). Unit: [Nm]. Default: -0.2 0.2.

Steering.Mech.Column.Damp.c = *value*

Specifies the column damping coefficient. Unit: [Nms/rad]. Default: 0.06.

Steering.Mech.Column.Damp.M_max = *value*

Specifies the upper and lower limit of column damping torque. Unit: [Nm]. Default: 0.1.

Steering.Mech.Column.NonUform.Kind = *Kind*

Specifies the kind of the column non-uniformity. Possible values are:

- "": No non-uniformity selected. This is the default.
- *Curve*: Non-uniformity is described by a non-linear 1D Look-up table.
- *DblCardan*: Non-uniformity is calculated depending on the position of the four column mount points using a double universal joint.
- *User*: Non-uniformity is calculated by an user implemented function, as shown in the following example:

Example In User.c:

```
static void MyNonUniformAngFunc (tNonUniformAngIF *IF, double dt)
{
     //place here the model
     IF->iUpp2Low = ...;   //ratio upper to lower column shaft []
     IF->Ang_Low  = ...;   //angle of lower column shaft [rad]
}
int User_Register (void) {
     //register the user function
     Set_UserNonUniformAngFunc (MyNonUniformAngFunc);
     return 0;
}
```

Steering.Mech.Column.NonUform.Curve: *Table[UpperColAng LowerColAng]*

Specifies the non-uniform relation between the upper and lower column angle. First column is the upper column angle [deg] (angle between the upper fork and the plane E_{ABC} from Figure 14.12), second column is the lower column angle [deg] (angle between the lower fork and the plane E_{BCD}). The both column angles must have a range between 0...180 deg:

Example

```
Steering.Mech.Column.NonUniformAng.Curve:
   0.0000     0.0000
   6.0000     6.7938
   ...
   168.0000   166.4540
   174.0000   173.2062
   180.0000   180.0000
```

Steering.Mech.Column.NonUform.DblCardan.Pnt[A, B, C, D] = *x y z*

Specifies the position of the four column mointing points A, B, C and D according the Figure 14.12.

Steering.Mech.Column.NonUform.DblCardan.ForkAng = *value*

Specifies the difference angle between the two fork planes in the intermediate axle. Corresponds to the parameter λ in the equation (EQ 99). Unit: [deg]. Default: 0.

Steering.Mech.Column.NonUform.DblCardan.UppAng_off = *value*

Specifies the offset angle δ_{uc}^{offset} of the first fork (upper column fork) in point B in Figure 14.12. Unit: [deg]. Default: 0.

Steering.Mech.Column.NonUform.DblCardan.NPlane = *bool*

Specifies the fork orientation in the first universal joint ABC. If the null plane orientation is chosen the first fork between the points A and B is parallel to the frame plane E_{ABC} and the second fork between the points B and C is perpendicular to the frame plane E_{ABC}, see Figure 14.12. Otherwise the first fork is perpendicular and the second fork is parallel to the frame plane. Default: 1 (null plane is taken).

Steering.Mech.TorsBar.k = *value*

Specifies the stiffness of the torsion bar. For EPS column model this stiffness represents k_{uc} in (EQ 94), otherwise this stiffness is combined with the stiffness of other parts to the total stiffness k_{lc} in (EQ 94). Unit: [Nm/deg]. Default: 2.

Steering.Mech.TorsBar.k: ***Table [Velocity Angle Torque]***

Specifies the torsion bar torque map. First column is the vehicle velocity [km/h]; second column is the torsion bar twist angle [deg] and the third column is the torsion bar twist torque [Nm]. The range for the twist angle must be between zero and the upper twist angle limit, described below.
Using this map the torsion bar stiffness is determined. For EPS column model this stiffness represents k_{uc} in (EQ 94), otherwise this stiffness is combined with the stiffness of other parts to the total stiffness k_{lc} in (EQ 94).

Steering.Mech.TorsBar.k.Amplify =*value*

Specifies the amplification factor of torsion bar torque map. Default: 1.0.

Steering.Mech.TorsBar.d = ***value***

Specifies the damping of the torsion bar. Unit: [Nms/rad]. Default: 0.6.

Steering.Mech.TorsBar.Fric.k = ***value***

Specifies the stiffness of the torsion bar exponential spring friction model. Corresponds to the parameter k_{ESF} in (EQ 96). Unit: [Nm/rad]. Default: 7000.

Steering.Mech.TorsBar.Fric.M_max = *lowerlimit upperlimit*

Specifies the upper and lower limit of the torsion bar exponential spring friction torque. Corresponds to the parameter F_{lim} in (EQ 95) and (EQ 96). Unit: [Nm]. Default: -0.01 0.01.

Steering.Mech.TorsBar.TwistAng_max =*value*

Specifies the upper and lower limit of the torsion bar twist angle. Unit: [deg]. Default: 5.

Steering.Mech.TorsBar.k_max =*value*

Specifies the stiffness of the torsion bar outside of the upper and lower twist angle.
Unit: [Nm/deg]. Default: 200.

Steering.Mech.Rack.mass = ***value***

Specifies the steering rack mass including the steering rod mass. Is a part of total steering mass m_R besides the wheel and suspension generalized mass. Unit: [kg]. Default: 3.

Steering.Mech.Rack.travel = *value*

Specifies the maximum steering rack travel from zero position ($s_R = 0$) to a end stop. The travel in both directions left and right is supposed to be the same. At the reached end of the travel a stiff spring damper element limits the rack motion. Unit: [m]. Default: 0.08.

Steering.Mech.Rack.Fric.k = *value*

Specifies the stiffness of the rack exponential spring friction model. Corresponds to the parameter k_{ESF} in (EQ 96). Unit: [N/m]. Default: 8.0e6.

Steering.Mech.Rack.Fric.F_max = *lowerlimit upperlimit*

Specifies the upper and lower limit of the rack exponential spring friction force. Corresponds to the parameter F_{lim} in (EQ 95) and (EQ 96). Unit: [N]. Default: -75 75.

Steering.Mech.Rack.Fric.cP = *value*

Specifies the friction force increase due to pressure, which is added to the force $F_{F,R}$. Used only with HPS. Unit: [N/bar]. Default: 13.

Steering.Mech.Rack.Damp.c = *value*

Specifies the rack damping coefficient. Unit: [Ns/m]. Default: 550.

Steering.Mech.Rack.Damp.F_max = *limit*

Specifies the upper and lower limit of the rack damping force. Unit: [N]. Default: 25.

Steering.Mech.Fric.Amplify = *value*

Amplification factor of all exponential spring friction elements. The values for the exponential spring friction is generally obtained from a steering test bench. Unit: []. Default: 1.0.

Steering.Mech.Mesh.k = *value*

Specifies the stiffness of the steering gear mesh. This stiffness is combined with the stiffness of other parts to the total stiffness k_{lc} in (EQ 94).
Unit: [N/m]. Default: 4e6.

Steering.Mech.HardyD.k = ***value***

Specifies the stiffness of the hardy disk. This stiffness is combined with the stiffness of other parts to the total stiffness k_{lc} in (EQ 94). Unit: [Nm/deg]. Default: 3.

Steering.Mech.SteerByTorque =*bool*

Specifies the kind of steering input for IPGDriver. Default: 1 (steer by torque).

General Parameters for Power Assistance Module

Steering.PAM.Kind = ***Kind***

Specifies the kind of power assistance calculation. Possible values are:

- *HPS*: Power assistance calculated by the hydraulic module. This is the default.
- *EPSapa_s*: Power assistance calculated by the simplified electrical module EPSapa.
- *EPSapa_e*: Power assistance calculated by the extensive electrical module EPSapa.
- *EPSc_s*: Power assistance calculated by the simplified electrical module EPSc.
- *EPSc_e*: Power assistance calculated by the extensive electrical module EPSc.
- *DVA*: The desired power assistance is set using Direct Variable Access on quantity *Steer.AssistFrc, Steer.AssistTrqCol* or *Steer.AssistTrqPin*.
- *User*: The desired power assistance is calculated by an user implemented function. The user can set the assistance force to the rack `IF->AssistFrc`, the assistance torque to the column `IF->AssistTrqCol` or to the pinion `IF->AssistTrqPin`. Following example shows the implementation for the assistance force to the rack:

Example
```
In User.c:
static void MyCalcAssistFrc (void *MP, tSteeringIF *IF, double dt)
{
    double AssistFrc;
    AssistFrc = ...; //place here the power assistance model
    IF->AssistFrc = AssistFrc;
}
int User_Register (void)
{
    Set_UserAssistFrcFunc(MyCalcAssistFrc); //register the user function
    return 0;
}
```

Parameters for Hydraulic Module

Steering.Hyd.Piston.Di = ***value***

Specifies the inner diameter of the piston. Needed for the calculation of the piston area A_P in (EQ 102). Unit: [m]. Default: 0.03.

Steering.Hyd.Piston.Do = ***value***

Specifies the outer diameter of the piston. Needed for the calculation of A_P in (EQ 102). Unit: [m]. Default: 0.05.

Steering.Hyd.Chamber.P_min =*value*

Specifies the system pressure in the piston without assistance. Unit: [bar]. Default: 1.0.

Steering.Hyd.Chamber.Cq = ***value***

Specifies the flow coefficient to/from chamber. Corresponds to the parameter C_q in (EQ 100). Unit: []. Default: 0.5.

Steering.Hyd.Chamber.A_Orf =*value*

Specifies the area of orifice flow to chamber. Corresponds to the parameter A_O in (EQ 100). Unit: [m^2]. Default: 8.0e-6.

Steering.Hyd.Cylinder.L = ***value***

Specifies the length of the cylinder. Unit: [m]. Default: 0.2.

Steering.Hyd.OilDensity = ***value***

Specifies the oil density. Corresponds to the parameter ς in (EQ 100).
Unit: [kg/m^3]. Default: 800.

Steering.Hyd.BulkModul = ***value***

Specifies the "effective" bulk modulus of the oil and its containments and pipe system. Corresponds to the parameter β_F in (EQ 101). Unit: [N/m^2]. Default: 3.0e7.

Steering.Hyd.PumpCap = ***value***

Specifies the pump capacity. Unit: [m^3/rad]. Default: 1.91e-7.

Steering.Hyd.P_drop = *value*

Specifies the HPS pressure loss along the hydraulic line (from the cylinder to the pump). Unit: [bar]. Default: 3.0.

Steering.Hyd.BoostKind = *Kind*

Specifies the hydraulic boost characteristic kind. Possible values are:

- *Curve*: Use a two-dimensional non-linear curve depending on torsion bar torque defined by the parameter *Steering.Hyd.BoostCurve* and *Steering.Hyd.Boost-Curve.Amplify* (Default).
- *Map*: Use a three-dimensional non-linear map depending on torsion bar torque and vehicle velocity defined by the parameter *Steering.Hyd.BoostMap* and *Steer-ing.Hyd.BoostMap.Amplify*.
- *DVA*: The desired boost characteristic is set using Direct Variable Access on quantity *Steer.Pfeffer.HPS.pDelta_stat* with the unit [Pa].
- *User*: The desired boost characteristic is calculated by an user implemented function, as shown in the following example:

Example `In User.c:`

```
static double MyCalcHydBoostFunc (tSteeringIF *IF, double TwistTrq, double dt)
{
     double DeltaPressure;
     DeltaPressure = ...; //place here the delta pressure calculation [Pa]
     return DeltaPressure;
}
int User_Register (void) {
     Set_UserHydBoostFunc(MyCalcHydBoostFunc); //register the user function
     return 0;
}
```

Steering.Hyd.BoostCurve: *Table [TorsionBarTorque Pressure]*

Specifies the hydraulic boost curve. First column is the torsion bar torque [Nm], second column is the target pressure difference between the two chambers [bar]. Corresponds to the Δp in the Figure 14.15.

Example

```
Steering.Hyd.BoostCurve:
   -8.0 60.0
   -7.0 30.0
   -6.0 16.0
   ...
   -3.0 4.0
   -2.0 1.0
   -1.0 0.0
   0.0 0.0
   1.0 0.0
   2.0 1.0
   3.0 4.0
   ...
   6.0 16.0
   7.0 30.0
   8.0 60.0
```

Steering.Hyd.BoostCurve.Amplify =*value*

Amplification factor of the hydraulic boost curve. Unit: []. Default: 1.0.

Steering.Hyd.BoostMap: *Table [Velocity TorsionBarTorque Pressure]*

Specifies the hydraulic boost map. First column is the velocity [km/h]; second column is the torsion bar torque [Nm]; third column is the target pressure difference between the two chambers [bar]. Corresponds to the Δp in the Figure 14.15.

Example

```
Steering.Hyd.BoostMap:
   0 -8.0 60.0
   0 -7.0 30.0
   ...
   0 7.0 30.0
   0 8.0 60.0
   ...
   150 -8.0 20.0
   150 -7.0 18.0
   ...
   150 8.0 20.0
   150 7.0 18.0
```

Steering.Hyd.BoostMap.Amplify =*value*

Amplification factor of the hydraulic boost map. Unit: []. Default: 1.0.

Steering.Hyd.WithPresLag = *bool*

Indicates if the pressure build-up in the both chambers is delayed with a time lag or not. 1=with time lag (1er order model); 0=no time lag (chamber pressure = target pressure). Default: 1.

Parameters for all Electrical Modules

Steering.Elec.Motor.Kt =*value*

Specifies the motor torque constant. Corresponds to the parameter K_t in (EQ 104). Unit: [Nm/A]. Default: 0.3.

Steering.Elec.RCBN2Motor.Ratio =*value*

Specifies the transmission ratio electric motor speed / recirculating ball nut speed. Corresponds to the parameter I_{belt} in (EQ 105). Unit: []. Default: 2.5.

Steering.Elec.BoostKind = *Kind*

Specifies the electrical boost characteristic kind. Possible values are:

- *Curve*: Use a two-dimensional non-linear curve depending on torsion bar torque defined by the parameter *Steering.Elec.BoostCurve* and *Steering.Elec.BoostCurve.Amplify*
- *Map*: Use a three-dimensional non-linear map depending on torsion bar torque and vehicle velocity defined by the parameter *Steering.Elec.BoostMap* and *Steering.Elec.BoostMap.Amplify* (Default).
- *DVA*: The desired boost characteristic is set using Direct Variable Access on quantity *Steer.Pfeffer.EPS.iA_stat* with the unit [A].
- *User*: The desired boost characteristic is calculated by an user implemented function, as shown in the following example:

Example In User.c:

```
static double MyCalcElecBoostFunc (tSteeringIF *IF, double TwstTrq, double dt)
{
     double MotorCurrent;
     MotorCurrent = ...; //place here the motor current calculation [A]
     return MotorCurrent;
}
int User_Register (void) {
     Set_UserElecBoostFunc(MyCalcElecBoostFunc); //register the user function
     return 0;
}
```

Steering.Elec.BoostCurve: *Table [TorsionBarTorque MotorCurrent]*

Specifies the electrical boost curve. First column is the torsion bar torque [Nm], second column is the motor current [A]. Corresponds to the i_M in the (EQ 105).

Example

```
Steering.Elec.BoostCurve:
   -4.2 20
   -4.1 15.1
   -3.9 10
   -3.5 5.1
   -2 1
   0 0
   2 1
   3.5 5.1
   3.9 10
   4.1 15.1
   4.2 20
```

Steering.Elec.BoostCurve.Amplify =*value*

Amplification factor of the electrical boost curve. Unit: []. Default: 1.0.

Steering.Elec.BoostMap: *Table [Velocity TorsionBarTorque MotorCurrent]*

Specifies the electrical boost map. First column is the vehicle velocity [km/h]; second column is the torsion bar torque [Nm]; third column is the motor current [A]. Corresponds to the i_M in the (EQ 105).

Example

```
Steering.Elec.BoostMap:
   0 -4.2 20
   0 -4.1 15.1
   ...
   0 4.1 15.1
   0 4.2 20
   ...
   150 -4.2 12
   150 -4.1 7
   ...
   150 4.1 7
   150 4.2 12
```

Steering.Elec.BoostMap.Amplify =*value*

Amplification factor of the electrical boost map. Unit: []. Default: 1.0.

Parameters for simplified Electrical Module (only EPSapa)

Steering.Elec.RCBN.Ratio =*value*

Specifies the transmission ratio of recirculating ball nut system: ball screw translation / ball nut rotation. Corresponds to the parameter I_{KGT} in (EQ 105). Unit: [m/rad]. Default: 1.6e-3.

Parameters for simplified Electrical Module (only EPSc)

Steering.Elec.AssistAtCol =*bool*

Specifies if the assisting torque of EPSc model is supplied at column or at the pinion.
1 means assisting at column, 0 means assisting at pinion. Default: 1.

Parameters for extensive Electrical Module (EPSapa and EPSc)

Following parameters are additional parameters to the simplified electrical modules.

Steering.Elec.Motor.I = *value*

Specifies the inertia of the motor. Corresponds to the parameter J_M in the (EQ 107).
Unit: [kgm^2]. Default: 1.5e-4.

Steering.Elec.Motor.ViscFricCoeff =*value*

Specifies the motor viscous friction coefficient. Corresponds to the parameter B in the (EQ 107). Unit: [Nms]. Default: 0.02.

Steering.Elec.CurrentCtrl.p = *value*
Steering.Elec.CurrentCtrl.i = *value*

Specifies the parameters of the current PI-Controller. Default: p=5, i=0.1;

Steering.Elec.Belt.k = *value*

Specifies the stiffness of the belt. Corresponds to the parameter k_{belt} in the (EQ 106).
Unit: [Nm/rad]. Default: 120.

Steering.Elec.Belt.d = ***value***

Specifies the damping of the belt. Corresponds to the parameter d_{belt} in the (EQ 106). Unit: [Nms/rad]. Default: 0.1.

Parameters for extensive Electrical Module (only EPSapa)

Steering.Elec.RCBN.mass = ***value***

Specifies the mass of the recirculating ball nut system. Corresponds to the parameter m_{bn} in the (EQ 111). Unit: [kg]. Default: 1.

Steering.Elec.RCBN.I = ***value***

Specifies the inertia of the recirculating ball nut system. Corresponds to the parameter J_{KGT} in the (EQ 109). Unit: [kgm^2]. Default: 3.0e-4.

Steering.Elec.RCBN.k = ***value***

Specifies the stiffness of the recirculating balls. Corresponds to the parameter k_{KGT} in the (EQ 110). Unit: [N/m]. Default: 1.0e7.

Steering.Elec.RCBN.d = ***value***

Specifies the damping of the recirculating balls. Corresponds to the parameter d_{KGT} in the (EQ 110). Unit: [Ns/m]. Default: 1.0e5.

Steering.Elec.RCBN.Fric.k = ***value***

Specifies the stiffness of the recirculating ball nut system exponential spring friction model. Corresponds to the parameter k_{ESF} in (EQ 96). Unit: [Nm/rad]. Default: 2.

Steering.Elec.RCBN.Fric.M_max = ***lowerlimit upperlimit***

Specifies the upper and lower limit of the recirculating ball nut system exponential spring friction torque. Corresponds to the parameter F_{lim} in (EQ 95) and (EQ 96).
Unit: [Nm]. Default: -0.02 0.02.

Steering.Elec.RCBN.Fric.kt = ***value***

Specifies the stiffness of the recirculating ball nut system translational exponential spring friction model. Corresponds to the parameter k_{ESF} in (EQ 96). Unit: [N/m]. Default: 1.0e6.

Steering.Elec.RCBN.Fric.F_max = ***lowerlimit upperlimit***

Specifies the upper and lower limit of the recirculating ball nut system translational exponential spring friction force. Corresponds to the parameter F_{lim} in (EQ 95) and (EQ 96). Unit: [N]. Default: -60 60.

Steering.Elec.Housing.k = ***value***

Specifies the stiffness between the ball nut and housing. Corresponds to the parameter k_{bn} in the (EQ 108). Unit: [N/m]. Default: 3.0e5.

Steering.Elec.Housing.d = ***value***

Specifies the damping between the ball nut and housing. Corresponds to the parameter d_{bn} in the (EQ 108). Unit: [Ns/m]. Default: 1.5e5.

Parameters for User PAM Modules (DVA or User)

Steering.Elec.AssistAtCol =*bool*

Specifies if the assisting torque of user specified model is supplied at column or at the pinion to consider the torsion bar at the right position: 1 means assisting at column, 0 means assisting at pinion. Default: 0.

14.6 Steering System "User C-code / Simulink Plug-in / FMU"

Following parameters are required for the initialization of the output variables in the interface struct *tSteeringCfgIF*:

Steering.SInputKind = *KindStr*

Optional. Specifies the supported input kind by the user steering model.
Following kinds are supported:

- *Angle*: Steering only by angle
- *Torque*: Steering only by torque
- *AngleAndTorque*: Steering by angle and torque

Steering.PrefByDriver = *KindStr*

Optional. Specifies the preferred input kind by the driver model.
Following kinds are supported:

- *Angle*: Steering by angle
- *Torque*: Steering by torque

14.7 User Accessible Quantities for Steering Systems

Please refer to section 24.9 'Steering System'.

Chapter 15
Powertrain

15.1 Overview

The powertrain which acts as a global system within the CarMaker VVE is divided into a number of subsystems with well defined interfaces. In the following the basic concept of the subsystems and interfaces is described.

The powertrain's subsystems can be assigned to three levels: Control strategy, control units and electro mechanic components (see Figure 15.1). PTControl contains a superordinated powertrain control strategy that mainly manages the powertrain's state (e.g. ignition on, drive) and the distribution of electrical and mechanic energy. Each control unit supervises one kind of electro mechanic components in order to establish the energy distribution planned by PTControl. The subsystems of the third level are models of the powertrain's electro mechanic components (such as motors, gearboxes, batteries etc.).

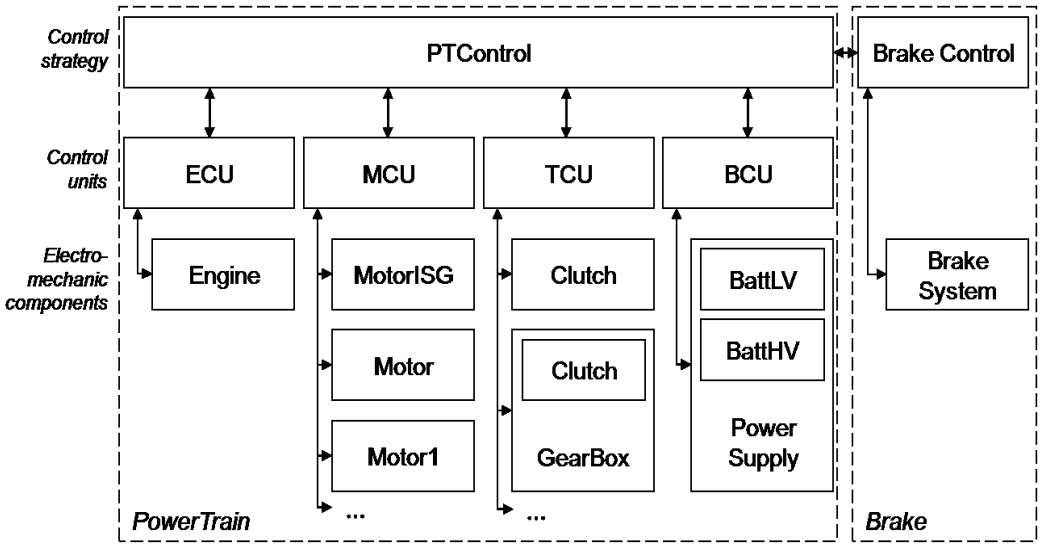


Figure 15.1: Structure of CarMaker powertrain

Every CarMaker powertrain model is built of subsystems of these three levels. The number of subsystems and their disposition depend on the powertrain's architecture. Each powertrain architecture combines engine, integrated starter generator (ISG), motors, clutches and gearboxes to *Drive Sources* that represents the torque creation and transmission to the driveline (input shaft of a differential or wheel). There can be between one and four Drive Sources per powertrain architecture.

15.2 PTControl

15.2.1 Overview

PTControl represents the superordinated powertrain control strategy. It determines target values for the control units (e.g. target torque for each motor) based on the inputs from DrivMan/VehicleControl and the current driving situation.

There are five tasks, that PTControl needs to fulfill for each powertrain architecture no matter if it is a conventional, hybrid or electric vehicle:

- Regulation of the vehicle operation state based on key position or activation of the start-stop-button
- Interpretation of the gas pedal position as torque demand
- Mechanic energy management: Determination of the current strategy mode and torque distribution to combustion engine and electric motors as defined by this strategy mode
- Battery management: Regulation of batteries' state of charge by assigning generator torques and regulating the exchange of electric power between different electric circuits
- Estimation of maximum possible and current regenerative braking torque at each wheel
- Calculation of characteristic values such as average fuel consumption

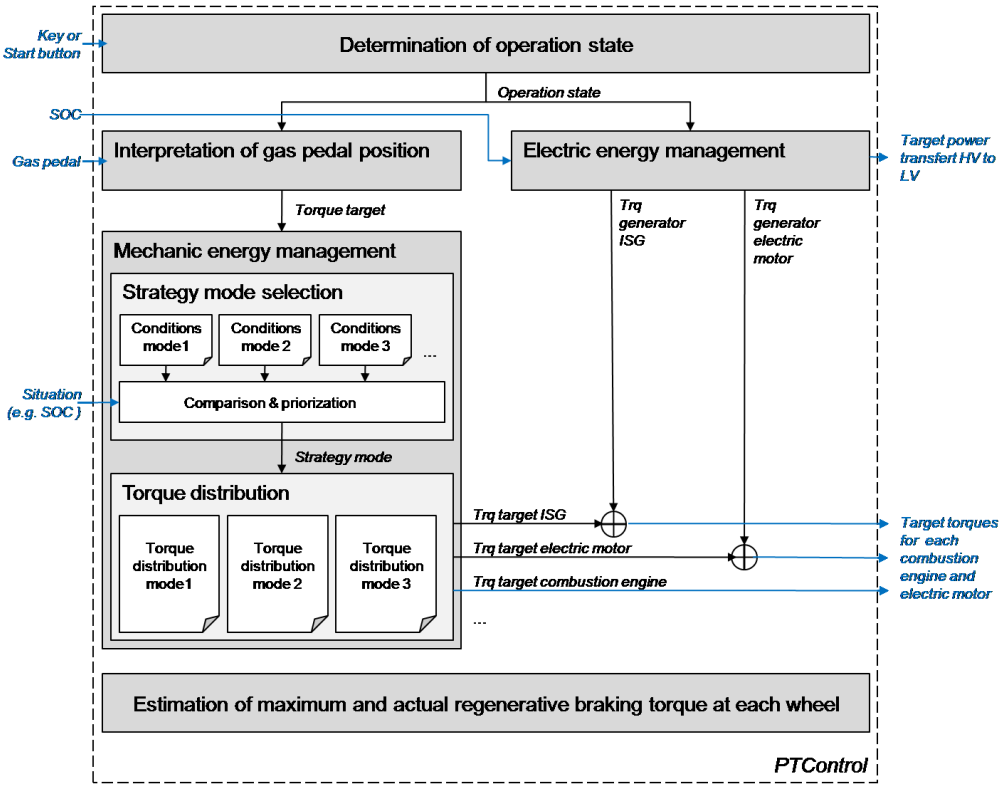


Figure 15.2: General structure of IPGPowertrain PTControl models

PowerTrain.Control.Kind = *KindStr VersionId*

Selection of PTControl model kind to use. The IPGPowertrain provides different PTControl models that can be used for the specified powertrain models.

ModelName	KindStr	Description
Generic	Generic	Control strategy for a conventional *Generic* powertrain
Micro	Micro	Control strategy for a conventional *Generic* powertrain with start-stop logic (micro hybrid)
Parallel P1	Parallel_P1	Control strategy for the *Parallel* powertrain (variant P1 with one clutch)
Parallel P2	Parallel_P2	Control strategy for the *Parallel* powertrain (variant P2 with two clutches)
Axle Split	AxleSplit	Control strategy for the *AxleSplit* powertrain
Power Split	PowerSplit	Control strategy for the *PowerSplit* powertrain
Serial	Serial	Control strategy for the *Serial* powertrain
Electrical	BEV	Control strategy for the *Electrical* powertrain

Powertrain control strategies are highly influenced by the powertrain's architecture. That is why PTControl is part of the powertrain model in CarMaker. One powertrain architecture can run with a multiplicity of control strategies, but one control strategy is compatible with only a few powertrain architectures. For this reason the description of the PTControl models is in the corresponding powertrain module (see section 15.7 'Powertrain Models').

The powertrain architecture - in particular the number and disposition of motors and electric circuits - has a huge impact on PTControl interface and determines the limits of electric and mechanic energy management. In other words: It is impossible to run all conventional, hybrid and electric powertrains with the same control strategy. Nevertheless, the PTControl models of CarMaker are built based on some basic modules. The sum of all these modules is described in section 'Powertrain'.

15.2.2 PTControl Interface

The interface between PTControl and the control units also depends on the powertrain architecture - especially on the number of motors and gearboxes. The following sections deal with the superset of the interface's quantities, which correspond to the global PTControl interface defined in the headerfile *PowerTrain.h*.

There are two interface structs defined for the information exchange between the PTControl model and the rest of the powertrain model and the simulation program, one for the initialization and one for the evaluation function.

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tPTControlCfgIF*:

Input Variable	Unit	Description
PTKind		Powertrain kind (generic, parallel, electric, serial)
ClKind		Clutch kind (friction, converter)
nMotor		Number of motors (without ISG)
nGearBoxM		Number of gearboxes (without gearbox for engine)

Input Variable	Unit	Description
nWheels		Number of vehicle wheels
WheelRadius	m	Mean wheel radius
Aero_Frcx	N	Aerodynamic drag force with velocity=1m/s
DriveSourcePos[*ds*]		Position where drive source *ds* is applied
DL_iDiff_mean		Driveline mean differential ratio
BattLV.SOC_min BattHV.SOC_min	%	Low / high voltage battery minimum admitted state of charge
BattLV.SOC_max BattHV.SOC_max	%	Low / high voltage battery maximum admitted state of charge [%]
BattLV.Capacity BattHV.Capacity	Ah	Low / high voltage battery capacity
BattLV.Voltage BattHV.Voltage	V	Open circuit voltage of low / high voltage battery
Engine.rotv_idle	rad/s	Engine idle speed
Engine.rotv_max	rad/s	Engine maximum speed
Engine.rotv_opt	rad/s	Engine optimum speed with optim. consumption
Engine.Fuel_l2kWh	kWh/l	Factor for fuel conversion to kWh
Engine.TrqFull	Nm	1D-Lookup table for engine full load torque
Engine.TrqDrag	Nm	1D-Lookup table for engine drag torque
Engine.TrqOpt	Nm	1D-Lookup table for engine torque with optim. consumption
ISG.Level Motor<*m*>.Level		PowerSupply voltage level (LV, HV1, HV2) of integrated starter generator / electric motor *m*
ISG.Ratio Motor<*m*>.Ratio		Ratio between motor shaft and driven shaft of integrated starter generator / electric motor *m*
ISG.rotv_Mot_max Motor<*m*>.rotv_Mot_max	rad/s	Maximum motor rotation speed of integrated starter generator / electric motor *m*
ISG.rotv_Gen_max Motor<*m*>.rotv_Gen_max	rad/s	Maximum generator rotation speed of integrated starter generator / electric motor *m*
ISG.TrqMot_max Motor<*m*>.TrqMot_max	Nm	1D-Lookup table for maximum motor torque of integrated starter generator / electric motor *m*
ISG.TrqGen_max Motor<*m*>.TrqGen_max	Nm	1D-Lookup table for maximum generator torque of integrated starter generator / electric motor *m*
GearBox.GBKind GearBoxM<*gb*>.GBKind		Gearbox kind (manual, automatic)
GearBox.ClKind GearBoxM<*gb*>.ClKind		Gearbox clutch kind (friction, converter)
GearBox.nFGears GearBoxM<*gb*>.nFGears		Gearbox number of forward gears
GearBox.iFGear[*gear*] GearBoxM<*gb*>.iFGear[gear]		Gearbox ratios of forward gears
GearBox.nBGears GearBoxM<*gb*>.nBGears		Gearbox number of backward gears
GearBox.iBGear[*gear*] GearBoxM<*gb*>.iBGear[gear]		Gearbox ratios of backward gears

Input Variable	Unit	Description
PlanetGear.Ratio<*pg*>		Planet gear ratio (ring/sun radius); only for PowerSplit architecture

Output Variable	Unit	Description
StartEngineWithSST		1: Powertrain is activated with start-stop button 0: Powertrain is activated with key only
Velocity_max	m/s	Vehicle maximum possible velocity for instruments

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tPTControlIF*:

Input Variable	Unit	Description
Key	-	Vehicle key position
SST	bool	Activation start-stop button
SelectorCtrl	-	Gearbox selector control
Clutch	-	Clutch pedal position
Brake	-	Brake pedal position
Gas	-	Gas pedal position
Velocity	m/s	Vehicle speed
GearNoTrg	-	Gearbox target gear (to overwrite TCU)
UserSignal<*us*>	-	User defined signals *us*
ECU_Status	-	Engine control unit status
EngineIn.Engine_on	bool	Engine on flag
EngineIn.rotv	rad/s	Current engine rotation speed
EngineIn.Trq	Nm	Current engine torque
EngineIn.TrqDrag	Nm	Engine drag torque
EngineIn.TrqFull	Nm	Engine full load torque
EngineIn.TrqOpt	Nm	Engine torque with optimal consumption
EngineIn.FuelFlow	l/s	Current engine fuel flow
MCU_Status	-	Motor control unit status
ISGIn.rotv MotorIn<*m*>.rotv	rad/s	Current rotation speed of integrated starter generator / electric motor *m*
ISGIn.Trq MotorIn<*m*>.Trq	Nm	Current motor torque of integrated starter generator / electric motor *m*
ISGIn.TrqMot_max MotorIn<*m*>.TrqMot_max	Nm	Maximum motor torque on driven shaft of integrated starter generator / electric motor *m*
ISGIn.TrqGen_max MotorIn<*m*>.TrqGen_max	Nm	Maximum generator torque on driven shaft of integrated starter generator / electric motor *m*
ISGIn.i_M2W<*pos*> MotorIn<*m*>.i_M2W<pos>	-	Theoretical ratio from integrated starter generator / electric motor *m* to wheel (without considering friction clutch)

Input Variable	Unit	Description
ISGIn.PwrElec MotorIn<*m*>.PwrElec	W	Electric power of integrated starter generator / electric motor *m*
TCU_Status	-	Transmission control unit status
ClutchIn.Pos	-	Clutch position
ClutchIn.rotv_in	rad/s	Clutch input shaft rotation speed
ClutchIn.rotv_out	rad/s	Clutch output shaft rotation speed
ClutchIn.Trq_in	Nm	Clutch input shaft torque
ClutchIn.Trq_out	Nm	Clutch output shaft torque
ClutchIn.i_TrqIn2Out	-	Ratio between clutch input shaft torque and output shaft torque
GearBoxIn.GearNo GearBoxM_In<*gb*>.GearNo GearBoxIn.GearNo_dis GearBoxM_In<gb>.GearNo_dis	-	Gearbox current gear
GearBoxIn.i GearBoxM_In<*gb*>.i	-	Gearbox current gear ratio
GearBoxIn.Trq_DriveSrc_trg GearBoxM_In<*gb*>.Trq_DriveSrc_trg	Nm	Optional drive source target torque from TCU to PTControl (e.g. while shifting)
GearBoxIn.rotv_in GearBoxM_In<*gb*>.rotv_in	rad/s	Gearbox input shaft rotation speed
GearBoxIn.rotv_out GearBoxM_In<*gb*>.rotv_out	rad/s	Gearbox output shaft rotation speed
GearBoxIn.Trq_in GearBoxM_In<*gb*>.Trq_in	Nm	Gearbox input shaft torque
GearBoxIn.Trq_out GearBoxM_In<*gb*>.Trq_out	Nm	Gearbox output shaft torque
GearBoxIn.Clutch.Pos GearBoxM_In<*gb*>.Clutch.Pos GearBoxIn.Clutch_dis.Pos GearBoxM_In<*gb*>.Clutch_dis.Pos	-	Gearbox clutch position
GearBoxIn.Clutch.rotv_in GearBoxM_In<*gb*>.Clutch.rotv_in GearBoxIn.Clutch_dis.rotv_in GearBoxM_In<*gb*>.Clutch_dis.rotv_in	rad/s	Gearbox clutch input shaft rotation speed
GearBoxIn.Clutch.rotv_out GearBoxM_In<*gb*>.Clutch.rotv_out GearBoxIn.Clutch_dis.rotv_out GearBoxM_In<*gb*>.Clutch_dis.rotv_out	rad/s	Gearbox clutch output shaft rotation speed
GearBoxIn.Clutch.Trq_in GearBoxM_In<*gb*>.Clutch.Trq_in GearBoxIn.Clutch_dis.Trq_in GearBoxM_In<*gb*>.Clutch_dis.Trq_in	Nm	Gearbox clutch input shaft torque
GearBoxIn.Clutch.Trq_out GearBoxM_In<*gb*>.Clutch.Trq_out GearBoxIn.Clutch_dis.Trq_out GearBoxM_In<*gb*>.Clutch_dis.Trq_out	Nm	Gearbox clutch output shaft torque

Input Variable	Unit	Description
GearBoxIn.Clutch.i_TrqIn2Out GearBoxM_In<*gb*>.Clutch.i_TrqIn2Out GearBoxIn.Clutch_dis.i_TrqIn2Out GearBoxM_In<*gb*>.Clutch_dis.i_TrqIn2Out	-	Ratio between gearbox clutch input shaft torque and output shaft torque
BCU_Status	-	Battery control unit status
BattLVIn.SOC BattHVIn.SOC	%	Low / high voltage battery instantaneous state of charge
BattLVIn.SOH BattHVIn.SOH	%	Low / high voltage battery state of health
BattLVIn.Current BattHVIn.Current	A	Low / high voltage battery electric current
BattLVIn.AOC BattHVIn.AOC	Ah	Low / high voltage battery instantaneous amount of charge
BattLVIn.Temp BattHVIn.Temp	K	Low / high voltage battery temperature
BattLVIn.Energy BattHVIn.Energy	kWh	Low / high voltage battery remaining energy capacity
BattLVIn.Pwr_max BattHVIn.Pwr_max	W	Low / high voltage battery maximum charge/discharge power
PwrSupplyIn.Pwr_LV	W	Total electric power (generators and consumers) on low voltage electric circuit
PwrSupplyIn.Pwr_HV1 PwrSupplyIn.Pwr_HV2	W	Total electric power (generators and consumers) on high voltage 1 / 2 electric circuit
PwrSupplyIn.Voltage_LV	V	Current voltage on low voltage electric circuit
PwrSupplyIn.Voltage_HV1 PwrSupplyIn.Voltage_HV2	V	Current voltage on high voltage 1 / 2 electric circuit
PwrSupplyIn.Pwr_HV1toLV	W	Current transferred electric power from high voltage 1 electric circuit to low voltage electric circuit
PwrSupplyIn.Pwr.HV1toHV2	W	Current transferred electric power from high voltage 1 electric circuit to high voltage 2 electric circuit
PwrSupplyIn.Pwr_HV1toLV_max	W	Maximum possible transferred electric power from high voltage 1 electric circuit to low voltage electric circuit
PwrSupplyIn.Pwr.HV1toHV2_max	W	Maximum possible transferred electric power from high voltage 1 electric circuit to high voltage 2 electric circuit
PwrSupplyIn.Eta_HV1toLV	-	DC/DC efficiency from high voltage 1 electric circuit to low voltage electric circuit
PwrSupplyIn.Eta_HV1toHV2	-	DC/DC efficiency from high voltage 1 electric circuit to high voltage 2 electric circuit
WheelIn<*wh*>.Trq_BrakeReg_trg	Nm	Target regenerative braking torque at wheel *wh* (absolute value) from brake control
DriveSrcIn[*ds*].Trq_trg	Nm	Target torque for drive source *ds* (e.g. intervention by brake control unit / ESP ECU)

Output Variable	Unit	Description
OperationState	-	Vehicle operation state
OperationError	-	Vehicle operation error or warning
Ignition	bool	Vehicle ignition
StrategyMode		Strategy mode (boost, assist, start/stop)
EngineOut.set_ISC	bool	Idle speed controller activated
EngineOut.FuelCutOff	bool	Flag if fuel is cut-off
EngineOut.Load *	-	Engine target load
EngineOut.Trq_trg *	Nm	Engine target torque
EngineOut.rotv_trg *	rad/s	Engine target rotation speed
ISGOut.Load * MotorOut<*m*>.Load *	-	Integrated starter generator / motor *m* target load
ISGOut.Trq_trg * MotorOut<*m*>.Trq_trg *	Nm	Integrated starter generator / motor *m* target torque
ISGOut.rotv_trg * MotorOut<*m*>.rotv_trg *	rad/s	Integrated starter generator / motor *m* target rotation speed
ClutchOut.Pos *	-	Clutch target position
ClutchOut.rotv_out_trg *	rad/s	Clutch output shaft target rotation speed
ClutchOut.Trq_out_trg *	Nm	Clutch output shaft target torque
GearBoxOut.GearNoTrg * GearBoxM_Out<*gb*>.GearNoTrg * GearBoxOut.GearNoTrg_dis * GearBoxM_Out<*gb*>.GearNoTrg_dis *	-	Gearbox target gear
GearBoxOut.set_parkBrake GearBoxM_Out<*gb*>.set_ParkBrake	bool	Gearbox park brake set
GearBoxOut.i_trg * GearBoxM_Out<*gb*>.i_trg *	bool	Gearbox target ratio
GearBoxOut.rotv_in_trg * GearBoxM_Out<*gb*>.rotv_in_trg *	rad/s	Gearbox input shaft target rotation speed
GearBoxOut.Trq_out_trg * GearBoxM_Out<gb>.Trq_out_trg *	Nm	Gearbox output shaft target torque
GearBoxOut.Clutch.Pos * GearBoxM_Out<*gb*>.Clutch.Pos * GearBoxOut.Clutch_dis.Pos * GearBoxM_Out<*gb*>.Clutch_dis.Pos *	-	Gearbox clutch target position
GearBoxOut.Clutch.rotv_out_trg * GearBoxM_Out<*gb*>.Clutch.rotv_out_trg * GearBoxOut.Clutch_dis.rotv_out_trg * GearBoxM_Out<*gb*>.Clutch_dis.rotv_out_trg *	rad/s	Gearbox clutch output shaft target rotation speed

Output Variable	Unit	Description
GearBoxOut.Clutch.Trq_out_trg * GearBoxM_Out<*gb*>.Clutch.Trq_out_trg * GearBoxOut.Clutch_dis.Trq_out_trg * GearBoxM_Out<*gb*>.Clutch_dis.Trq_out_trg *	Nm	Gearbox clutch output shaft target torque
PwrSupplyOut.Pwr_HV1toLV_trg *	W	Target transferred electric power from high voltage 1 electric circuit to low voltage electric circuit
WheelOut<*wh*>.Trq_BrakeReg	Nm	Estimated current regenerative braking torque at wheel *wh* (absolute value)
WheelOut<*wh*>.Trq_BrakeReg_max	Nm	Estimated maximum possible regenerative braking torque at wheel *wh* (absolute value)

Please note:

- PTControl may provide **either** torques, loads **or** rotation speeds as target values for engine and motor control. Only one target signal may be set at a time! The signals that are not provided by PTControl are set to -99999 (= "not set").
- If the output signals, that are marked with *in the table above are set to a value other then -99999 (="not set"), they overwrite the target values calculated by the corresponding control unit (ECU, MCU, TCU). E.g. the target gear defined by TCU can be overwritten by PTControl.

15.2.3 Interface to Brake Control

In CarMaker, PTControl does not take any decision about the distribution of braking torques between electric motor and hydraulic brake. Brake control adopts this task, as it's demanded by ECE-R13 for security reasons. That leads to the following responsibility assignment:

- PTControl estimates the current maximum regenerative braking torque at each wheel.
- Brake Control calculates a target regenerative braking torque at each wheel based on maximum regenerative torques (see section 16.3.1 'Hydraulic Brake Control').
  In combination with an ESP ECU it is possible to set a target drive source torque.
- PTControl transforms these target regenerative torques into target torques for electric motors and evaluates the current regenerative braking torque at each wheel.
- Brake Control pilots the brake system based on these current regenerative braking torques in order to have the desired total braking torque at each wheel.

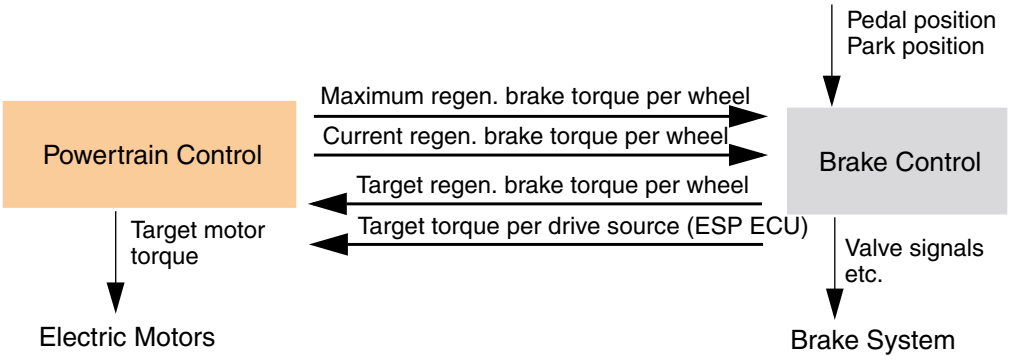


Figure 15.3: Interface between PTControl and Brake Control

Please notice: PTControl and Brake Control exchange the absolute values of the regenerative braking torques. That means all torques at their interface are positive.

IPGPowertrain PTControl models estimate the maximum admissible regenerative braking torque at each wheel $T_{BrakeRegMax}$ and the current regenerative braking torque at each wheel $T_{BrakeReg}$ the following way

$$T_{BrakeRegMax} = T_{GeneratorMax_{ISG}} \cdot i_{M2W_{ISG}} + \sum_{m=0}^{n} (T_{GeneratorMax_m} \cdot i_{M2W_m}) \quad \textbf{(EQ 122)}$$

$$T_{BrakeReg} = T_{Generator_{ISG}} \cdot i_{M2W_{ISG}} + \sum_{m=0}^{n} (T_{Generator_{mj}} \cdot i_{M2W_m}) \quad \textbf{(EQ 123)}$$

with the maximum generator torque $T_{GeneratorMax}$, the current generator torque $T_{Generator}$ and the current transmission ratio from motor to wheel i_{M2W} for each integrated starter generator *ISG* and motor *m*.

15.2.4 Operation states

Turning the ignition key of a vehicle or pressing its start button switches the operation state of the powertrain. PTControl models and DrivMan know five operation states, which correspond to different electric system and powertrain activation levels:

- Absent: Driver is outside the vehicle, power off
- PowerOff: Driver is in the vehicle, power off
- PowerAccessory: Electric power for accessory on
- PowerOn: Electric power on (vehicle ignition)
- Driving: Ready for driving or driving

PowerTrain.Control.StartEngineWithSST = *bool*

Control element for switching operation states: 1 = switch operation states with start button, 0 = switch operation states with key only. Default: 0.

Ignition key position or start button activation lead to transition between the operation states. The *VehicleOperator* in DrivMan module inputs these actions to PTControl in order to reach a desired operation state.

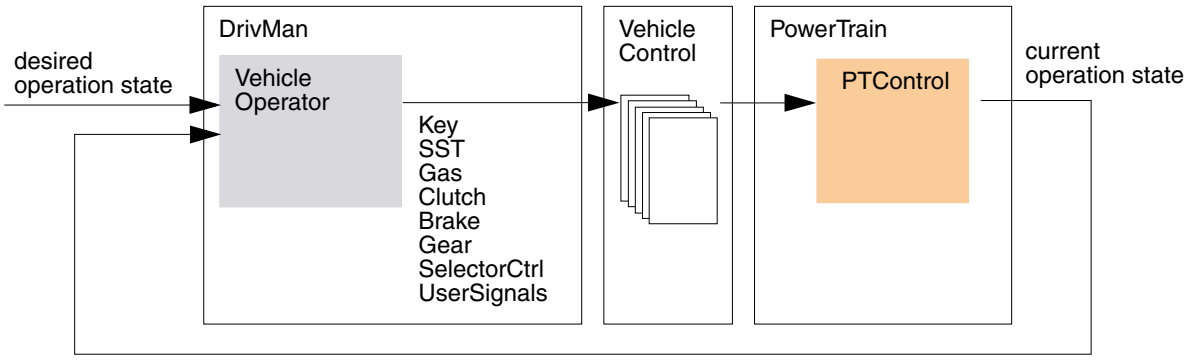


Figure 15.4: Interaction between *VehicleOperator* and *PTControl*

Operation State Machine

In PTControl models, delivered with IPGPowertrain, the determination of the current operation state is done by an integrated module, called *PTControl Operation State Machine* (PTControlOSM). This module determines, depending on the driver inputs (SST, key, gas, brake, etc.), which PTControl function is actually active to be called by PTControl (e.g.: function to get the powertrain to power off state or to start the powertrain). The return value of the current function is given as an input to the PTControlOSM module in the next step to detect an error or to know that the function has finished.

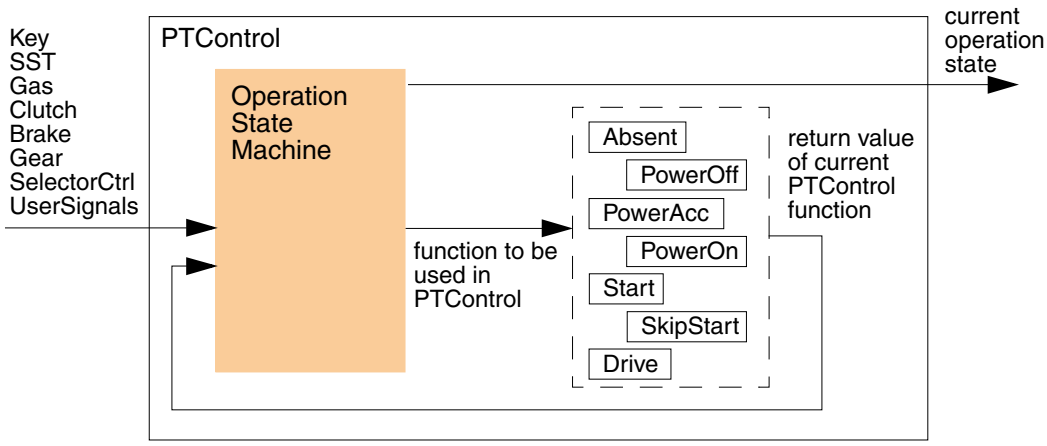


Figure 15.5: *Operation State Machine* within *PTControl*

PowerTrain.ControlOSM.Kind = *KindStrVersionId*

Selection of PTControl operation state machine model kind to use. The IPGPowertrain provides following model to use.

ModelName	KindStr	Description
Generic	Generic	Operation state switch with start-stop button or ignition key

The following table represents the input and output variables of the initialization interface struct *tPTControlOSM_CfgIF*:

Input Variable	Unit	Description
StartEngineWithSST	bool	1: Powertrain is activated with start-stop button 0: Powertrain is activated with key only
GBKind		Gearbox kind (manual, automatic)
Func_used<*fn*>	bool	Flag if PTControl model has a function *fn* to be called

The following table represents the input and output variables of the evaluation interface struct *tPTControlOSM_IF*:

Input Variable	Unit	Description
Key	-	Vehicle key position
SST	bool	Activation start-stop button
SelectorCtrl	-	Gearbox selector control
Clutch	-	Clutch pedal position
Brake	-	Brake pedal position
Gas	-	Gas pedal position
Velocity	m/s	Vehicle speed
GearNoTrg	-	Gearbox target gear
UserSignal<*us*>	-	User defined signals *us*
FuncReturn		Return value of current PTControl function: -1: error 0: normal calculation, not finished 1: normal calculation, finished

Output Variable	Unit	Description
OperationState		Vehicle operation state
OperationError		Vehicle operation error or warning
OSMFunc		Function to be used by PTControl model: 0: None, 1: Absent, 2: PowerOff 3: PowerAccessory, 4: PowerOn, 5: Start 6: SkipStart, 7: Drive

PTControl OSM Model "Generic"

The IPGPowertrain default model *Generic*, whose functionality is derived from the starting procedures of serial-production vehicles, supports the powertrain start using a key or a start-stop button SST. Below the flow process chart of the both variants are demonstrated.

Start with key

If the vehicle has only a conventional ignition key for starting, DrivMan and PTControl differentiate five key positions:

- KeyOut: Key is not in the ignition lock
- KeyIn_PowerOff: Key is in the ignition lock at initial position
- KeyIn_PowerAcc: Key is in the ignition lock, key is turned to position 1
- KeyIn_PowerOn: Key is in the ignition lock, key is turned to position 2
- KeyIn_Starter: Key is in the ignition lock, key is turned to position 3

Figure 15.6 and Table 15.1: show the key position and additional actions required for the transition between these operation states.

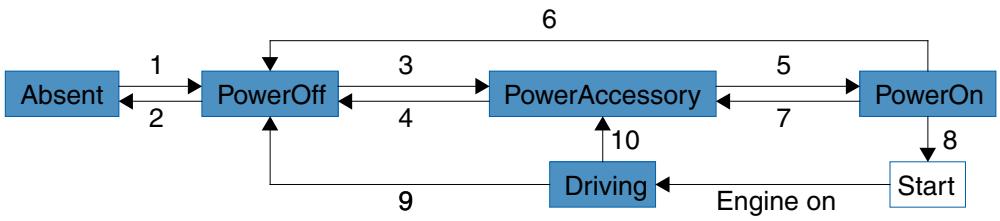


Figure 15.6: Switching operation states with key

No.	Transition	KeyOut	KeyIn_PowerOff:	KeyIn_PowerAccessory	KeyIn_PowerOn	KeyIn_Starter	Additional actions required
1	Absent to PowerOff		x	x	x	x	
2	PowerOff to Absent	x					
3	PowerOff to PowerAccessory			x	x	x	
4	PowerAccessory to PowerOff	x	x				
5	PowerAccessory to PowerOn				x	x	
6	PowerOn to PowerOff	x	x				
7	PowerOn to PowerAccessory			x			
8	PowerOn to Driving (via Start)					x	Pedals
9	Driving to PowerOff (in standstill or emergency off)	x	x				
10	Driving to PowerAccessory			x			

Table 15.1: Switching operation states with key

Start with button

If the vehicle has a start button, a succession of pressing this button and additional actions leads to state transitions. Switching to the operation states *PowerAccessory*, *PowerOn* and *Driving* is possible only if the key is inside the vehicle (that means, if key position is not *KeyOut*). Figure 15.7 and Table 15.2: show the actions required for the transition between these operation states for a vehicle with button (SST).

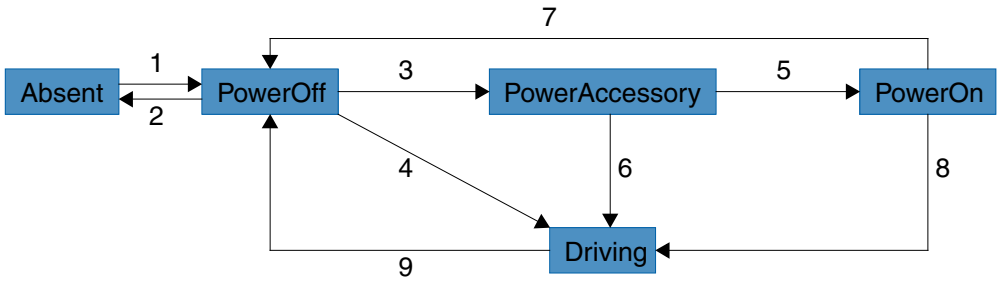


Figure 15.7: Switching operation states with button

No.	Transition	SST	Key	Additional actions required
1	Absent to PowerOff		≠ KeyOut	
2	PowerOff to Absent		= KeyOut	
3	PowerOff to PowerAccessory	x		
4	PowerOff to Driving	x		Pedals
5	PowerAccessory to PowerOn	x		
6	PowerAccessory to Driving	x		Pedals
7	PowerOn to PowerOff	x		
8	PowerOn to Driving	x		Pedals
9	Driving to PowerOff (in standstill)	x		Pedals
9	Driving to PowerOff (emergency off)	x		

Table 15.2: Switching operation states with button

Pedals

Just as in most real vehicles, the transition to the operation state *Driving* requires additional actions on the pedals and the gear selection. They depend on the gearbox type used:

- Manual gearbox:

$$\begin{gathered}\text{VC.GearNo} = 0\ \& \\ \text{VC.Clutch} >= \text{PowerTrain.ControlOSM.Clutch_lim}\ \& \\ \text{VC.Brake} >= \text{PowerTrain.ControlOSM.Brake_lim}\end{gathered} \quad \textbf{(EQ 124)}$$

- Automatic gearbox:

$$\begin{gathered}\text{VC.SelectorCtrl} = 0\ \& \\ \text{VC.Brake} >= \text{PowerTrain.ControlOSM.Brake_lim}\end{gathered} \quad \textbf{(EQ 125)}$$

- No gearbox:

$$\text{VC.Brake} >= \text{PowerTrain.ControlOSM.Brake_lim} \quad \textbf{(EQ 126)}$$

The Minimaneuver Command *VhclOp* enables you to switch operation states in a comfortable way (see User's Guide, section C.1.4 'Action Commands').

PowerTrain.ControlOSM.Brake_lim = *value*

Optional. Brake position that is required to change into the operation state *Driving* (compare to section 'Pedals'). Setting this parameter to 0 can lead to ambiguous state transition conditions for vehicles started with a button and thus should be avoided! Default: 0.3

PowerTrain.ControlOSM.Clutch_lim = *value*

Optional. Clutch position that is required to change into the operation state *Driving* for vehicles with manual gearbox (compare to section 'Pedals'). Setting this parameter to 0 can lead to ambiguous state transition conditions for vehicles started with a button and thus should be avoided! Default: 0.9

PowerTrain.ControlOSM.Key.tOff2Acc = *value*

Optional. Expected period of time for activation of the ignition key to change from *PowerOff* to *PowerAccessory* state. Default: 0.3s.

PowerTrain.ControlOSM.Key.tAcc2On = *value*

Optional. Expected period of time for activation of the ignition key to change from *PowerAccessory* to *PowerOn* state. Default: 0.3s.

PowerTrain.ControlOSM.Key.tOn2Off = *value*

Optional. Expected period of time for activation of the ignition key to change from *PowerOn* to *PowerOff* state. Default: 0.3s.

PowerTrain.ControlOSM.SST.tOff2Acc = *value*

Optional. Expected period of time for activation of the start-stop button to change from *PowerOff* to *PowerAccessory* state. Default: 0.3s.

PowerTrain.ControlOSM.SST.tAcc2On = *value*

Optional. Expected period of time for activation of the start-stop button to change from *PowerAccessory* to *PowerOn* state. Default: 0.3s.

PowerTrain.ControlOSM.SST.tOn2Off = *value*

Optional. Expected period of time for activation of the start-stop button to change from *PowerOn* to *PowerOff* state. Default: 3.0s.

PowerTrain.ControlOSM.SST.tOff2Drive = *value*

Optional. Expected period of time for activation of the start-stop button to change from *PowerOff* to *Driving* state. Default: 0.8s.

PowerTrain.ControlOSM.SST.tAcc2Drive = *value*

Optional. Expected period of time for activation of the start-stop button to change from *PowerAccessory* to *Driving* state. Default: 0.5s.

PowerTrain.ControlOSM.SST.tOn2Drive = *value*

Optional. Expected period of time for activation of the start-stop button to change from *PowerOn* to *Driving* state. Default: 0.2s.

15.2.5 Interpretation of gas pedal position

PTControl contains a module that reads in the current gas pedal position and translates it into a desired torque. The mechanical energy management will then split this torque demand up between combustion engine and electric motors. In general, the reference point for the desired torque is the point where the torques are provided by all engines and motors sum up. Please, refer to section 15.7 'Powertrain Models' for the reference points of the implemented powertrain architectures.

PowerTrain.Control.GasInterpret.Mode = *KindStr*

The PTControl library provides the following submodels:

ModelName	KindStr	Description
Characteristic Value	Linear	Torque demand is linear to gas pedal position
1D Look-Up Table	Curve	Transformation of gas pedal position into torque demand using a characteristic curve depending on gas pedal position
2D Look-Up Table	Curve2D	Transformation of gas pedal position into torque demand using characteristic curves depending on gas pedal position and vehicle velocity
DVA	DVA	Torque demand is given via Direct Variable Access to the variable *PT.Control.GasInterpret.Trq_trg*

Example `PowerTrain.Control.GasInterpret.Mode = Linear`

Gas interpreter model "Linear"

In this model, the desired torque T_{target} is linear to the gas pedal position gas. The transformation factor results from the difference between maximum torque demand at full gas T_{Full} and minimum torque demand at zero gas T_{Zero}.

$$T_{target} = (T_{Full} - T_{Zero}) \cdot gas + T_{Zero} \quad \text{(EQ 127)}$$

PowerTrain.Control.GasInterpret.TrqFull = *value*

Maximum torque demand for full gas. Default: 185 Nm.

PowerTrain.Control.GasInterpret.TrqZero = *value*

Minimum torque demand for zero gas. Default: -10 Nm.

Gas interpreter model "Curve"

The gas pedal position is transformed into a torque demand using a characteristic curve. The desired torque T_{target} is calculated the following way. In this formula λ represents the normalized target torque at the current gas pedal position.

$$T_{target} = T_{Zero} + \lambda \cdot (T_{Full} - T_{Zero}) \quad \text{(EQ 128)}$$

PowerTrain.Control.GasInterpret.TrqFull = *value*

Maximum torque demand for full gas. Default: 185Nm.

PowerTrain.Control.GasInterpret.TrqZero = *value*

Minimum torque demand for zero gas. Default: -10Nm.

PowerTrain.Control.GasInterpret.NormTrq: *Table*

Characteristic curve to transform gas pedal position into normed target torque. Both columns must begin with zero and end with 1.

Syntax

Infofile table mapping with 2 column
<gas pedal> <normalized target torque>

Gas interpreter model "Curve2D"

The gas pedal position is transformed into a torque demand using characteristic curves. The desired torque T_{target} is calculated the following way. In this formula λ represents the normalized target torque at the current gas pedal position and current normalized vehicle speed.

$$T_{target} = T_{Zero} + \lambda \cdot (T_{Full} - T_{Zero}) \quad \text{(EQ 129)}$$

PowerTrain.Control.GasInterpret.TrqFull = *value*

Maximum torque demand for torque demand calculation. Default: 185Nm.

PowerTrain.Control.GasInterpret.TrqZero = *value*

Minimum torque demand for torque demand calculation. Default: -10Nm.

PowerTrain.Control.GasInterpret.VelAmp = *value*

Maximum vehicle speed for normalizing vehicle speed. Default: 120km/h.

PowerTrain.Control.GasInterpret.NormTrq: *Table*

Characteristic curve to transform gas pedal position and normalized verhicle speed into normed target torque.

Syntax

Infofile table mapping with 3 column
<normalized vehicle speed> <gas pedal> <normalized target torque>

15.2.6 Mechanical energy management - Strategy modes

The mechanical energy management determines the current strategy mode and splits the torque demand up between the combustion engine and electric motors. It is only active during operation state *Driving*.

Each strategy mode corresponds to a specific torque distribution between the combustion engine and the electric motors. The PTControl models distinguish up to nine different strategy modes and three transient modes:

- StartStop: Standstill, engine off
- RegBrake: Regenerative braking; electric motor is used as a generator to apply a brake torque (recuperation of kinetic energy)
- RegDrag: Regenerative drag; electric motor is used as a generator to apply an artificial drag torque, if gas pedal is released (recuperation of kinetic energy)
- Coasting: engine off and disengaged; electric motors off
- ElecDrive: Electric drive; Driving with electric motors only, engine off
- LoadShift: Combustion engine runs at optimum torque while electric motor (as generator) regenerates the part of the energy which is not needed for driving
- Assist: Combustion engine runs at optimum torque while electric motor (as motor) applies the difference to demanded torque
- Boost: Combustion engine runs at maximum torque while electric motor applies the difference to demanded torque
- EngineDrive: Driving with combustion engine only, electric motors off
- EngineStart (transition mode): Combustion engine start procedure
- EngineSync (transition mode): Combustion engine synchronization to powertrain rotation speed
- EngineStop (transition mode): Combustion engine shut down

The strategy modes form two groups: Strategy modes with combustion engine off (StartStop, RegBrake, RegDrag, Coasting and ElecDrive) and strategy modes with combustion engine on (LoadShift, Assist, Boost and EngineDrive). In Figure 15.8, the dashed line separates these two groups. The transient modes (grey) are located at the transition between the groups. They control the procedure of starting and stopping the combustion engine.

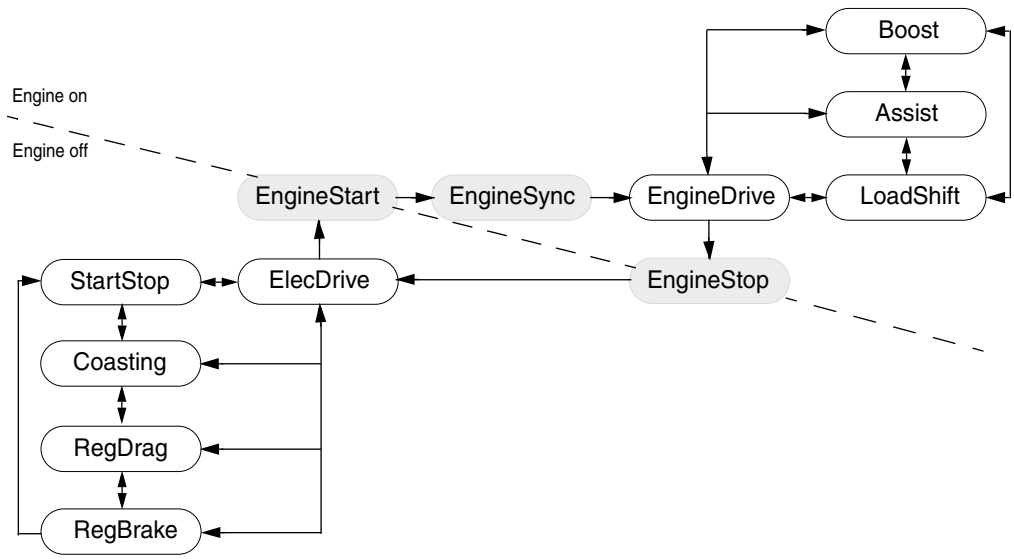


Figure 15.8: IPGPowertrain PTControl strategy modes - example Parallel P2

To avoid that the combustion engine is switched on or off for only a short time, the transition from one group to the other is not initiated instantaneous. Instead there is a certain observation time. During this time the last active strategy mode is executed, even if the strategy mode selection outputs a strategy mode of the other group as target mode.

PTControl calculates target torques for each electric motor and combustion engine in order to adapt a strategy mode. But for transition modes, it can provide a target rotation speed or load as well. For one engine or motor, there can only be one target value (torque, rotation speed or load) at a time. That means e.g., if PTControl outputs a target torque for an electric motor, target rotation speed and target load have to be set to -99999. This value is defined as "not set".

The availability of the strategy modes is closely linked to the powertrain's architecture. E.g. a pure electric powertrain is not able to execute the strategy mode EngineDrive - thus it is not part of its PTControl. Please, refer to the description of the PTControl models within the corresponding powertrain model section for a list of strategy modes available for each IPG-Powertrain model.

Besides, each strategy mode is submitted to specific conditions on vehicle speed, battery's state of charge, pedal positions and target torques. It is only executed, if these conditions are fulfilled. PTControl model distinguishes conditions that need to be fulfilled to enter a strategy mode and conditions that need to be fulfilled to stay in a strategy mode. Thus it avoids frequent changes between two modes and stabilizes the control strategy. If the conditions for several modes are fulfilled, the strategy mode with the highest priority is executed. For each PTControl model, a standard strategy mode is defined, which is executed when no other strategy mode is possible.

PT.Control.StrategyMode	Mode	Priority
0	StartStop	1
1	RegBrake	2
2	RegDrag	3
3	Coasting	4
4	ElecDrive	5
5	LoadShift	6
6	Assist	7
7	Boost	8
8	EngineDrive	9
9	EngineStart	0
10	EngineSync	0
11	EngineStop	0

The following paragraphs describe IPGPowertrain strategy modes sorted by their priority (high to low). Transient modes always have the highest priority.

Strategy mode "StartStop"

The strategy mode "StartStop" has the highest priority of all strategy modes except transient modes (priority 1). It corresponds to standstill with combustion engine off. During StartStop, the target torques for combustion engine and all electric motors are set to zero.

Figure 15.9 shows the conditions for entering StartStop (blue), if another strategy mode is active, and for staying in StartStop (green), if StartStop is currently active.

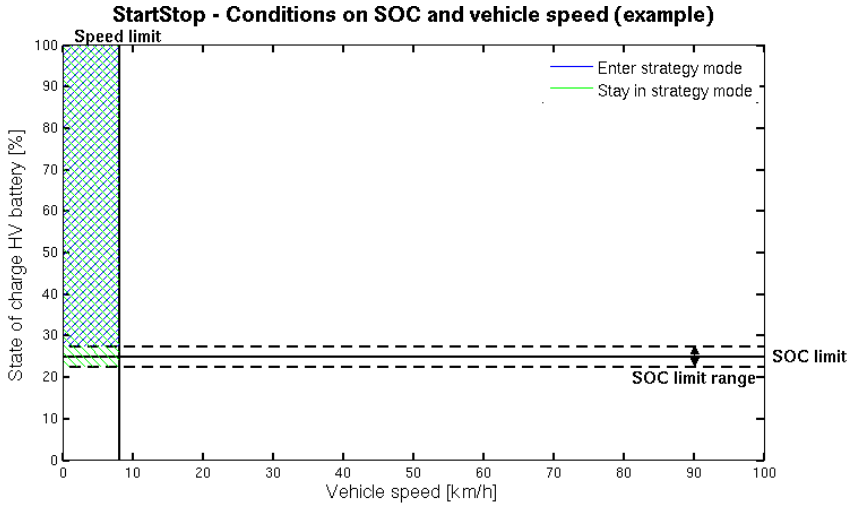


Figure 15.9: StartStop - conditions on SOC and vehicle speed

Additionally, conditions on brake, clutch and gas pedal positions are taken into account:

Quantity	Condition to enter StartStop	Condition to stay in StartStop
Brake	> Brake_limit + 0.5 Brake_limit_range	> Brake_limit - 0.5 Brake_limit_range
Clutch	> Clutch_limit - 0.5 Clutch_limit_range	> Clutch_limit + 0.5 Clutch_limit_range
Gas	< Gas_limit + 0.5 Gas_limit_range	< Gas_limit - 0.5 Gas_limit_range

Parameter StartStop

PowerTrain.Control.StartStop.Active = *bool*

Add strategy mode StartStop to PTControl model.

PowerTrain.Control.StartStop.SOC_lim = *value*

Lower boundary of battery's SOC (mean value) for strategy mode StartStop. Default: 25%.

PowerTrain.Control.StartStop.SOC_lim_range = *value*

Width of the transient zone around SOC lower boundary for strategy mode StartStop. Default: 5%.

PowerTrain.Control.StartStop.v_lim = *value*

Upper boundary of vehicle speed (mean value) for strategy mode StartStop. Default: 8km/h.

PowerTrain.Control.StartStop.Clutch_lim = *value*

Lower boundary of clutch pedal position (mean value) for strategy mode StartStop. Default: 0.95.

PowerTrain.Control.StartStop.Clutch_lim_range = *value*

Width of the transient zone around clutch pedal position's lower boundary for strategy mode StartStop. Default: 0.0.

PowerTrain.Control.StartStop.Gas_lim = *value*

Upper boundary of gas pedal position (mean value) for strategy mode StartStop. Default: 0.05.

PowerTrain.Control.StartStop.Gas_lim_range = *value*

Width of the transient zone around gas pedal position's upper boundary for strategy mode StartStop. Default: 0.0.

PowerTrain.Control.StartStop.Brake_lim = *value*

Lower boundary of brake pedal position (mean value) for strategy mode StartStop. Default: 0.075.

PowerTrain.Control.StartStop.Brake_lim_range = *value*

Width of the transient zone around brake pedal position's lower boundary for strategy mode StartStop. Default: 0.05.

Strategy mode "RegBrake"

The strategy mode RegBrake controls regenerative braking. It assigns generator torques to the electric motors to recuperate braking energy and thus recharge the battery instead of loosing energy for friction braking. This strategy mode has priority 2.

In CarMaker, PTControl does not take any decision about the distribution of braking torques between electric motor and conventional brake. Brake control unit BrakeControl adopts this task, as it's demanded by ECE-R13 for security reasons. The strategy mode RegBrake receives a target regenerative brake torque $T_{BrakeRegTarget_{pos}}$ for each wheel from BrakeControl. It calculates the electric motor's target torque $T_{EMtarget}$, that is necessary to approach these torques at the wheels without exceeding the demanded brake torques. In a conventional powertrain, the combustion engine's drag torque $T_{ICEdrag}$ acts as an additional brake torque. If in a hybrid powertrain the torque transmission between combustion engine and wheels is interrupted (e.g. disengaged clutch open), RegBrake adds this drag torque to the target regenerative torque for the electric motors.

$$T_{EMtarget} = -1 \cdot Min\left(\frac{T_{BrakeRegTarget_{pos}}}{i_{M2W_{pos}}}\right) + T_{ICEdrag} \qquad \textbf{(EQ 130)}$$

Figure 15.10 and Figure 15.11 show the conditions for entering RegBrake (blue), if another strategy mode is active, and for staying in RegBrake (green), if RegBrake is currently active.

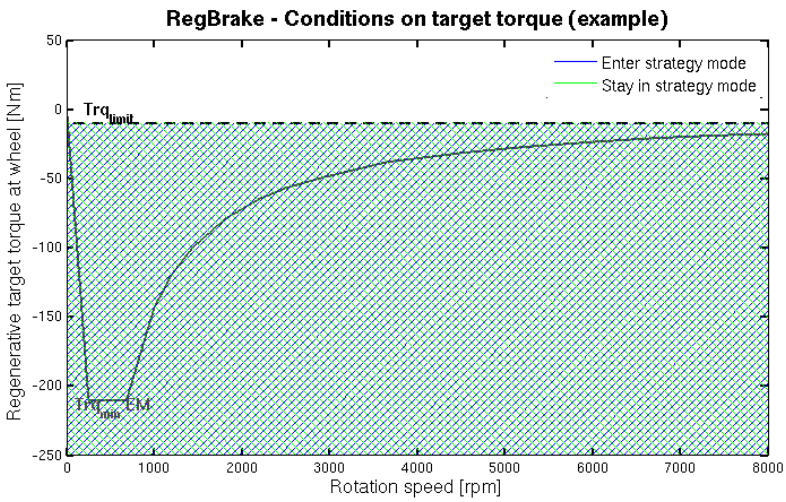


Figure 15.10: RegBrake - Conditions on target torque

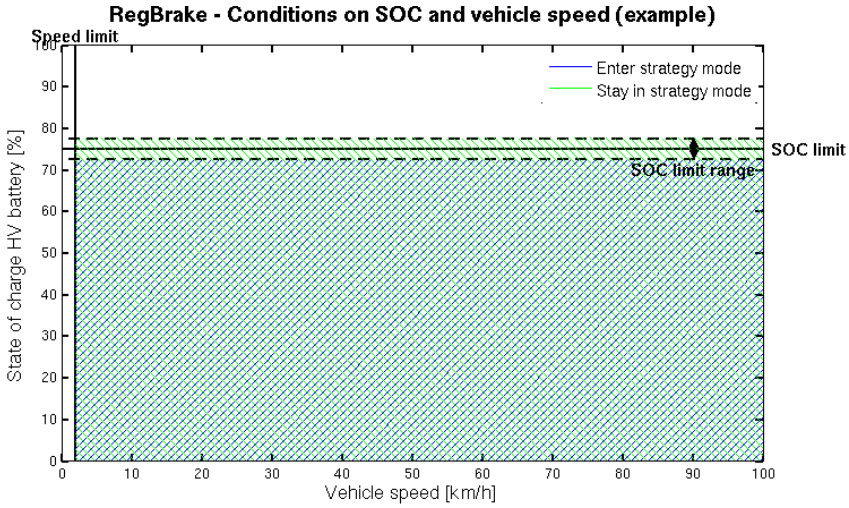


Figure 15.11: RegBrake - Conditions on SOC and vehicle speed

Parameter RegBrake

PowerTrain.Control.RegBrake.Active = *bool*

Add strategy mode RegBrake to PTControl model.

PowerTrain.Control.RegBrake.SOC_lim = *value*

Upper boundary of battery's SOC (mean value) for strategy mode RegBrake. Default: 75%.

PowerTrain.Control.RegBrake.SOC_lim_range = *value*

Width of the transient zone around SOC upper boundary for strategy mode RegBrake. Default: 5%.

PowerTrain.Control.RegBrake.v_lim = *value*

Lower boundary of vehicle speed (mean value) for strategy mode RegBrake.
Default: 2km/h.

PowerTrain.Control.RegBrake.Trq_RegBrake_lim = *value*

Lower boundary of target regenerative torque at each wheel (absolute value) for strategy mode RegBrake. Default: 2Nm.

Strategy mode "RegDrag"

The strategy mode RegDrag controls regenerative drag, also called 'e-sailing', and has priority 3. During this mode, an artificial drag torque is created using the electric motor as a generator. Thus, the powertrain's batteries are recharged instead of dragging the combustion engine. RegDrag derives the target torque for the electric motors from the (negative) torque demand from gas pedal position interpretation. The combustion engine is turned off and its torque transition to the wheels is disconnected (e.g. disengaged clutch open).

Figure 15.12 and Figure 15.13 show the conditions for entering RegDrag (blue), if another strategy mode is active, and for staying in RegDrag (green), if RegDrag is currently active.

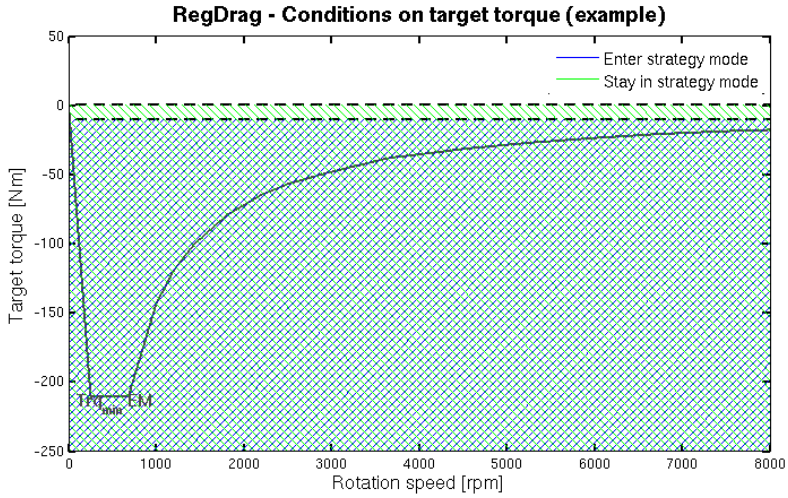


Figure 15.12: RegDrag - Conditions on target torque

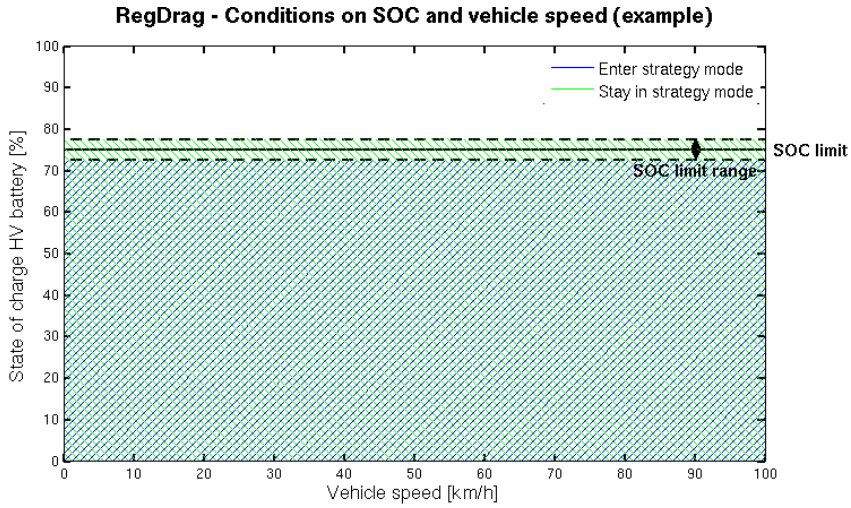


Figure 15.13: RegDrag - Conditions on SOC and vehicle speed

Parameter RegDrag

PowerTrain.Control.RegDrag.Active = *bool*

Add strategy mode RegDrag to PTControl model.

PowerTrain.Control.RegDrag.SOC_lim = *value*

Upper boundary of battery's SOC (mean value) for strategy mode RegDrag. Default: 75%.

PowerTrain.Control.RegDrag.SOC_lim_range = *value*

Width of the transient zone around SOC upper boundary for strategy mode RegDrag. Default: 5%.

PowerTrain.Control.RegDrag.v_lim = *value*

Lower boundary of vehicle speed only for entering in the strategy mode RegDrag.
For staying in the strategy mode RegDrag the velocity is not considered.
Default: 5km/h.

PowerTrain.Control.RegDrag.Trq_RegDrag_lim = *value*

Upper boundary of target torque (negative) for staying in strategy mode RegDrag (absolute value). Default: 2Nm.

Strategy mode "Coasting"

The strategy mode Coasting controls driving without torques, also called 'sailing', and has priority 4. During this mode, if the driver take foot off the gas pedal, the combustion engine is turned off and its torque transition to the wheels is disconnected (e.g. disengaged clutch open). The electric motor torques are also all set to zero.

Figure 15.14 shows the conditions for entering Coasting (blue), if another strategy mode is active, and for staying in Coasting (green), if Coasting is currently active.

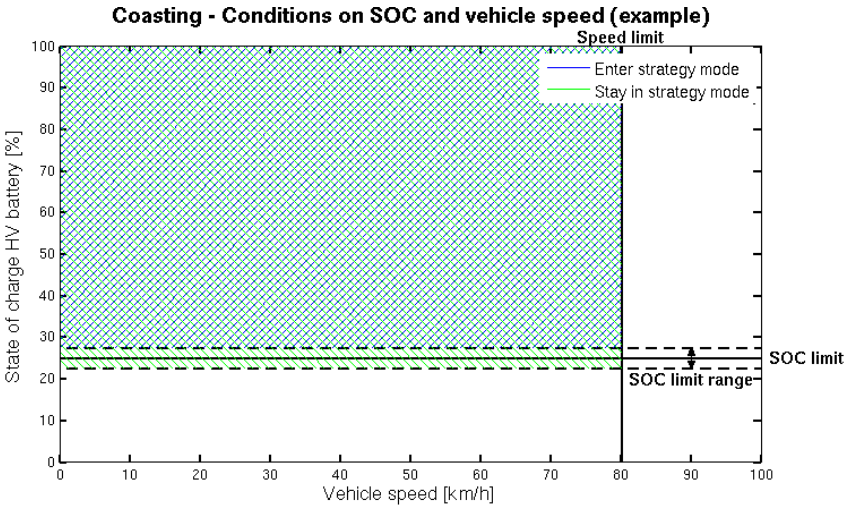


Figure 15.14: Coasting - conditions on SOC and vehicle speed

Additionally, condition on gas pedal position is taken into account:

Quantity	Condition to enter Coasting	Condition to stay in Coasting
Gas	< Gas_limit + 0.5 Gas_limit_range	< Gas_limit - 0.5 Gas_limit_range

Parameter Coasting

PowerTrain.Control.Coasting.Active = *bool*

Add strategy mode Coasting to PTControl model.

PowerTrain.Control.Coasting.v_lim = *value*

Upper boundary of vehicle speed (mean value) for strategy mode Coasting.
Default: 80km/h.

PowerTrain.Control.Coasting.SOC_lim = *value*

Lower boundary of battery's SOC (mean value) for strategy mode Coasting. Default: 25%.

PowerTrain.Control.Coasting.SOC_lim_range = *value*

Width of the transient zone around SOC lower boundary for strategy mode Coasting. Default: 5%.

PowerTrain.Control.Coasting.Gas_lim = *value*

Upper boundary of gas pedal position (mean value) for strategy mode Coasting.
Default: 0.05.

PowerTrain.Control.Coasting.Gas_lim_range = *value*

Width of the transient zone around gas pedal position's upper boundary for strategy mode Coasting. Default: 0.0.

Strategy mode "ElecDrive"

The strategy mode ElecDrive corresponds to pure electric driving and has the priority 5. The entire torque demand is attributed to the electric motors. Combustion engine is off, its target torque is zero and the torque transmission from engine to wheel is interrupted if possible (e.g. disengaged clutch open).

Figure 15.15 and Figure 15.16 show the conditions for entering ElecDrive (blue), if another strategy mode is active, and for staying in ElecDrive (green), if ElecDrive is currently active.

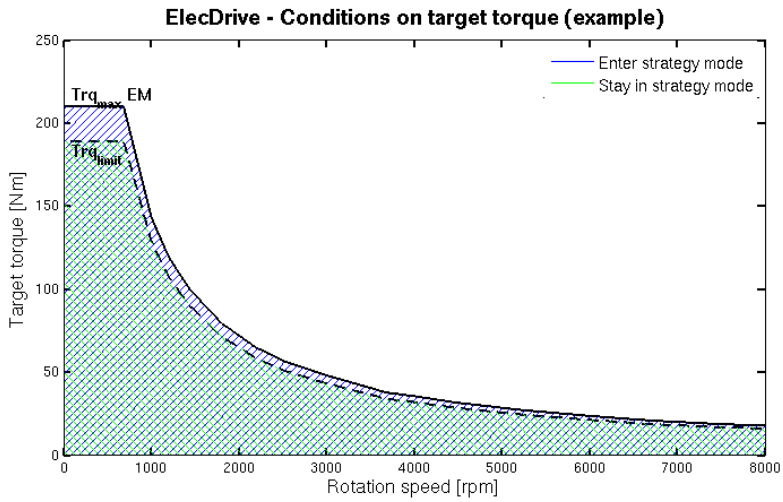


Figure 15.15: ElecDrive - Conditions on target torque

Here the torque limit in is calculated by

$$T_{limit} = \lambda \cdot T_{EMmax} \quad \textbf{(EQ 131)}$$

with $\lambda < 1$. Thus the conditions for staying in ElecDrive are narrower than for entering ElecDrive. This is due to the special location of ElecDrive on the boarder between modes with combustion engine on and modes with combustion engine off. If ElecDrive is currently active and the target torque passes over T_{limit}, the target strategy mode switches to a mode with engine on (EngineDrive, LoadShift, Assist or Boost). But it will take some time to start the engine (observation time + transient mode EngineStart). The electric motor should bridge this gap and therefore needs a torque reserve.

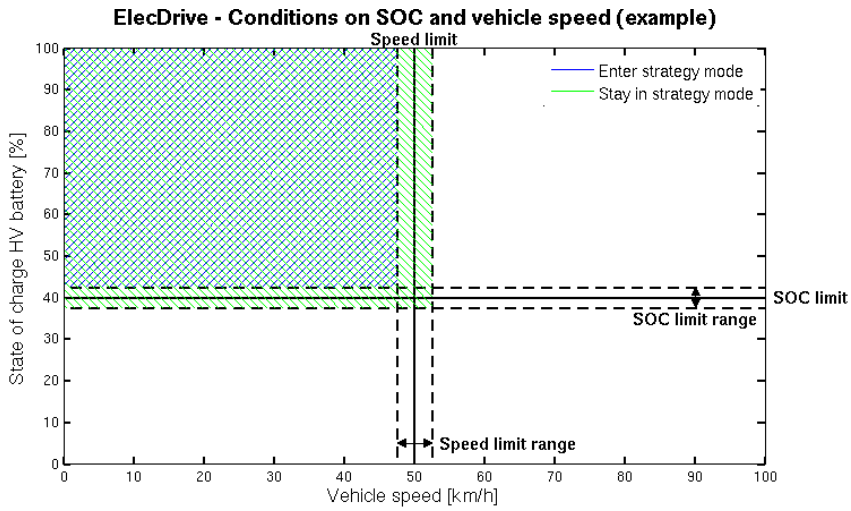


Figure 15.16: ElecDrive - Conditions on SOC and vehicle speed

Parameter ElecDrive

PowerTrain.Control.ElecDrive.Active = *bool*

Add strategy mode ElecDrive to PTControl model.

PowerTrain.Control.ElecDrive.SOC_lim = *value*

Lower boundary of battery's SOC (mean value) for strategy mode ElecDrive. Default: 40%.

PowerTrain.Control.ElecDrive.SOC_lim_range = *value*

Width of the transient zone around SOC lower boundary for strategy mode ElecDrive. Default: 5%.

PowerTrain.Control.ElecDrive.v_lim = *value*

Upper boundary of vehicle speed (mean value) for strategy mode ElecDrive.
Default: 50km/h.

PowerTrain.Control.ElecDrive.v_lim_range = *value*

Width of the transient zone around vehicle speed upper boundary for strategy mode ElecDrive. Default: 5km/h.

PowerTrain.Control.ElecDrive.TrqFac_lim_out = *value*

Part of maximum electric motor torque for target torque upper boundary for staying in strategy mode ElecDrive. Default: 90%.

Strategy mode "LoadShift"

If LoadShift is active, the target torque is smaller than the combustion engine's optimum torque at its current rotation speed. The optimum torque is the engine's torque for which fuel consumption is at its minimum (for current rotation speed). LoadShift sets the combustion engine's target torque to its optimum torque T_{ICEopt}. This means, the combustion engine generates more energy than necessary for driving. The surplus of energy is used for recharging the battery by assigning an appropriated generator torque to the electric motor. This strategy mode has priority 6.

$$T_{ICEtarget} = T_{ICEopt} \quad \text{(EQ 132)}$$

$$T_{EMtarget} = T_{target} - T_{ICEopt} < 0Nm \quad \text{(EQ 133)}$$

Figure 15.17 and Figure 15.18 show the conditions for entering LoadShift (blue), if another strategy mode is active, and for staying in LoadShift (green), if LoadShift is currently active.

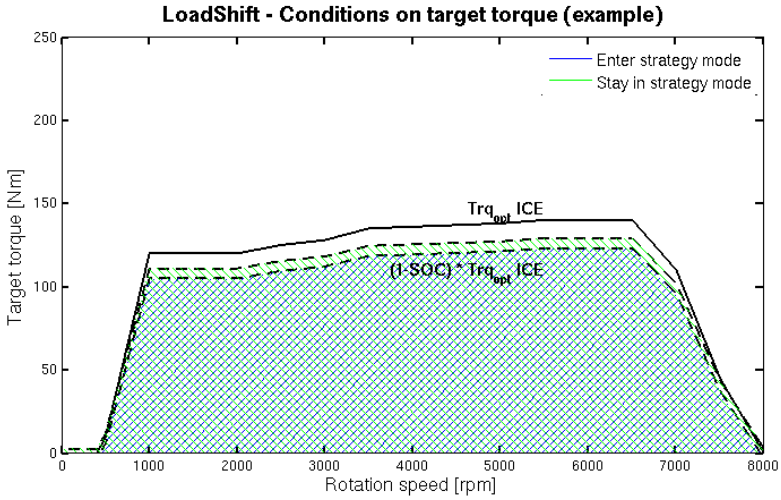


Figure 15.17: LoadShift - Conditions on target torque

Here the torque limit T_{limit} depends on the battery's state of charge:

$$T_{limit} = (1 - SOC) \cdot T_{ICEopt} \mp T_{range} \cdot 0.5 \quad \text{(EQ 134)}$$

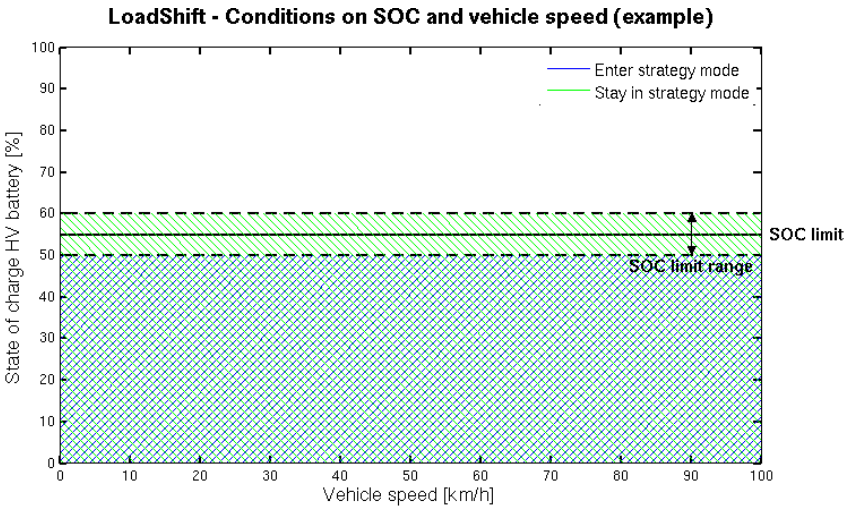


Figure 15.18: LoadShift - Conditions on SOC and vehicle speed

Additionally, the engines rotation speed has to be superior to its idle speed.

Parameter LoadShift

PowerTrain.Control.LoadShift.Active = *bool*

Add strategy mode LoadShift to PTControl model.

PowerTrain.Control.LoadShift.SOC_lim = *value*

Upper boundary of battery's SOC (mean value) for strategy mode LoadShift. Default: 55%.

PowerTrain.Control.LoadShift.SOC_lim_range = *value*

Width of the transient zone around SOC upper boundary for strategy mode LoadShift. Default: 10%.

PowerTrain.Control.LoadShift.Trq_lim_range = *value*

Width of the transient zone around target torque upper boundary for strategy mode LoadShift. Default: 4Nm.

Strategy mode "Assist"

If the strategy mode Assist is active, the target torque is bigger than the combustion engine's optimum torque and smaller than its maximum torque at the current rotation speed. The optimum torque is the engine's torque for which fuel consumption is at its minimum (for current rotation speed). Assist sets the combustion engine's target torque to its optimum torque T_{ICEopt}. It assigns the difference between target and optimum torque to the electric motor. This strategy mode has priority 7.

$$T_{ICEtarget} = T_{ICEopt} \quad \text{(EQ 135)}$$

$$T_{EMtarget} = T_{target} - T_{ICEopt} > 0Nm \quad \text{(EQ 136)}$$

Figure 15.19 and Figure 15.20show the conditions for entering Assist (blue), if another strategy mode is active, and for staying in Assist (green), if Assist is currently active.

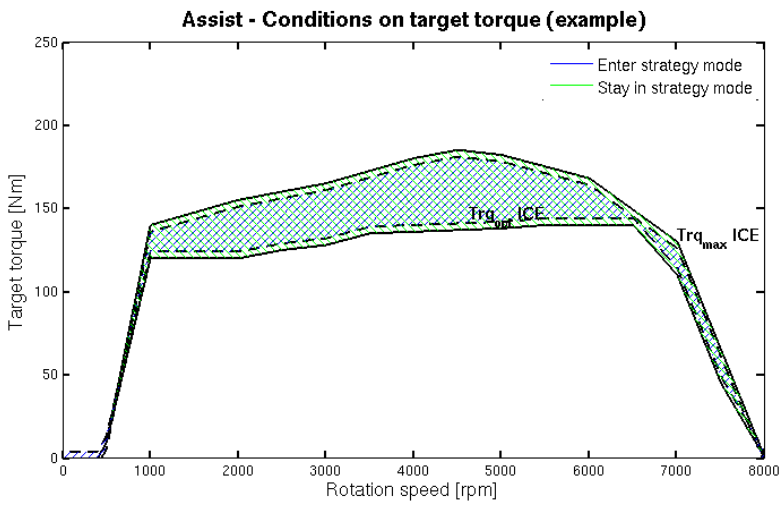


Figure 15.19: Assist - Conditions on target torque

For the upper and lower torque boundary a transient zone with the width T_{range} is used for a proper strategy mode change from and to the Assist mode.

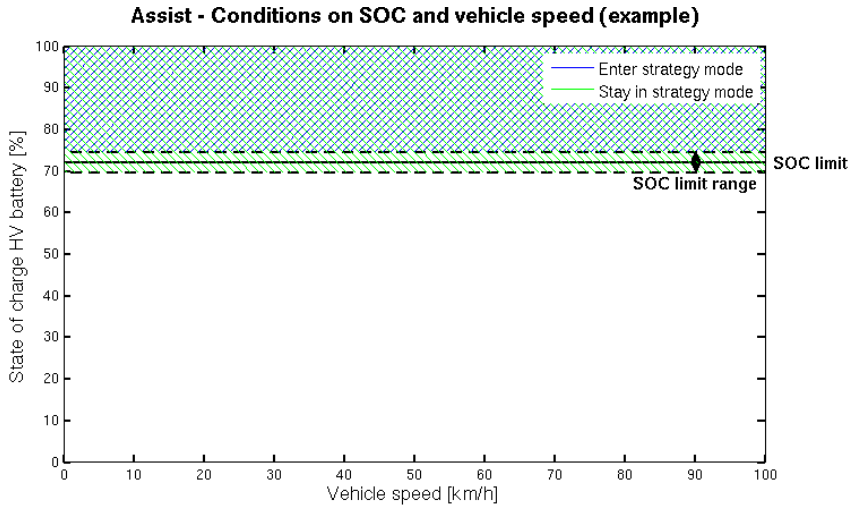


Figure 15.20: Assist - Conditions on SOC and vehicle speed

Parameter Assist

PowerTrain.Control.Assist.Active = *bool*

Add strategy mode Assist to PTControl model.

PowerTrain.Control.Assist.SOC_lim = *value*

Lower boundary of battery's SOC (mean value) for strategy mode Assist. Default: 72%.

PowerTrain.Control.Assist.SOC_lim_range = *value*

Width of the transient zone around SOC lower boundary for strategy mode Assist. Default: 5%.

PowerTrain.Control.Assist.Trq_lim_range = *value*

Width of the transient zone around target torque upper and lower boundary for strategy mode Assist. Default: 4Nm.

Strategy mode "Boost"

If the strategy mode Boost is active, the target torque is bigger than the combustion engine's maximum torque at the current rotation speed. Boost sets the combustion engine's target torque to engines maximum torque T_{ICEmax} and assigns the difference between target torque and maximum torque to the electric motor. This mode has the priority 8.

$$T_{ICEtarget} = T_{ICEmax} \quad \text{(EQ 137)}$$

$$T_{EMtarget} = T_{target} - T_{ICEmax} > 0Nm \quad \text{(EQ 138)}$$

Figure 15.21 and Figure 15.22 show the conditions for entering Boost (blue), if another strategy mode is active, and for staying in Boost (green), if Boost is currently active.

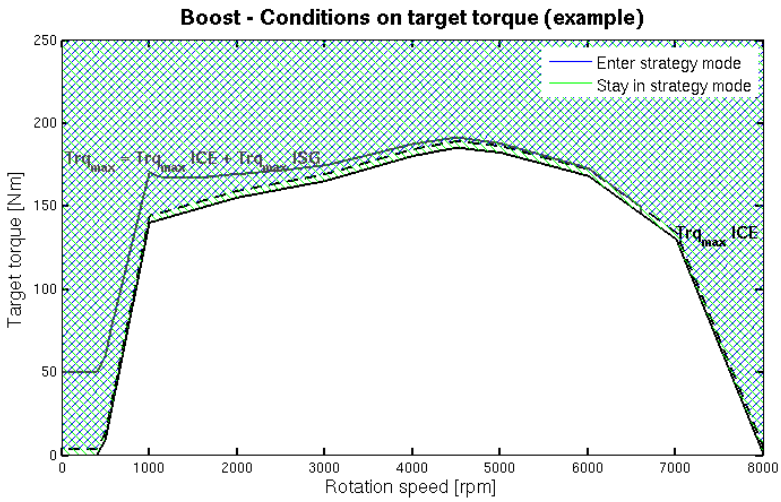


Figure 15.21: Boost - Conditions on target torque

For the lower torque boundary a transient zone with the width T_{range} is used for a proper strategy mode change from and to the Boost mode.

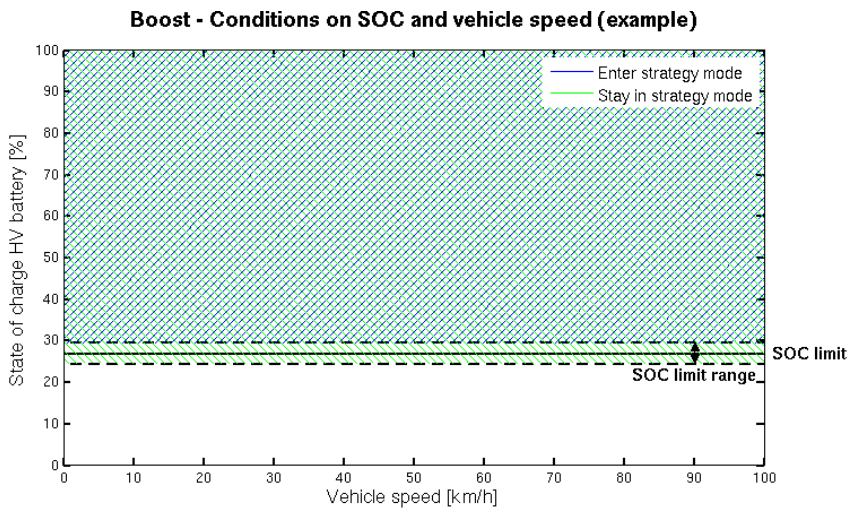


Figure 15.22: Boost - Conditions on SOC and vehicle speed

Parameter Boost

PowerTrain.Control.Boost.Active = *bool*

Add strategy mode Boost to PTControl model.

PowerTrain.Control.Boost.SOC_lim = *value*

Lower boundary of battery's SOC (mean value) for strategy mode Boost. Default: 27%.

PowerTrain.Control.Boost.SOC_lim_range = *value*

Width of the transient zone around SOC lower boundary for strategy mode Boost. Default: 5%.

PowerTrain.Control.Boost.Trq_lim_range = *value*

Width of the transient zone around target torque lower boundary for strategy mode Boost. Default: 4Nm.

Strategy mode "EngineDrive"

The strategy mode EngineDrive corresponds to pure combustion engine driving and has the priority 9. The entire torque demand is attributed to the combustion engine. Electric motor's target torque is zero.

EngineDrive is the standard strategy mode for conventional and hybrid vehicles, that is executed if the conditions of no other strategy mode are fulfilled. That is why it has no conditions and can not be deactivated.

Transient mode "EngineStart"

The mode EngineStart is located at the transition from strategy modes with engine off (StartStop, RegBrake, RegDrag, Coasting and ElecDrive) to strategy modes with engine on (LoadShift, Assist, Boost and EngineDrive). It pilots the combustion engine's start procedure by assigning maximum load to the starter motor until combustion engine is on (rotation speed > minimum engine rotation speed). If the powertrain contains an additional electric motor that is not used as starter motor, the entire target torque is assigned to this motor (which corresponds to the strategy mode ElecDrive).

To enter EngineStart, the following conditions have to be fulfilled for a specific observation time:

- Combustion engine is off during the current strategy mode
- Combustion engine should be on for the target strategy mode

EngineStart is active until the combustion engine runs (rotation speed > minimum engine rotation speed).

Parameter EngineStart

PowerTrain.Control.EngStart.tObsv = *value*

Observation time before entering the transition mode EngineStart. Default: 0.2s.

Transient mode "EngineSync"

The mode EngineSync is located at the transition from strategy modes with engine off to strategy modes with engine on. It pilots the synchronization and connection of the combustion engine to the powertrain, if the powertrain's rotation speed exceeds the combustion engine's idle speed. This situation may occurs, if combustion engine and electric motor are separated by an additional clutch (e.g. P2 parallel hybrid) or are located on different axles (e.g. AxleSplit hybrid).

EngineSync consists of three sequenced phases:

- Phase 1: The combustion engine's target speed is set to electric motor's current rotation speed, until the clutch's input shaft and output shaft have the same rotation speed. The entire target torque is assigned to the electric motor (which corresponds to the strategy mode ElecDrive).
- Phase 2: The clutch between combustion engine and electric motor closes with a user defined gradient. The combustion engine's target rotation speed is set to electric motor's rotation speed. The entire target torque is assigned to the electric motor (which corresponds to the strategy mode ElecDrive).
- Phase 3: If the clutch is closed, the combustion engine's target torque is increased with an user defined gradient. The electric motor's target torque is reduced in order to obtain the desired torque at the gearbox input shaft. This task is executed until the electric motor's target torque is zero.

EngineSync is executed directly after the transient mode EngineStart. It is active until phase 3 is completed.

Parameter EngineSync

PowerTrain.Control.EngSync.Grad_Clutch = *value*

Gradient for closing the clutch between engine and electric motor (phase2). Default: $10s^{-1}$.

PowerTrain.Control.EngSync.Grad_EngTrq = *value*

Gradient for transferring the target torque from electric motor to combustion engine (phase 3). Default: 500Nm/s.

Transient mode "EngineStop"

The mode EngineStop is located at the transition from strategy modes with engine on (LoadShift, Assist, Boost and EngineDrive) to strategy modes with engine off (StartStop, RegBrake, RegDrag, Coasting and ElecDrive). It pilots the combustion engine's stop procedure by assigning a braking torque to the starter motor until the engine's rotation speed is zero. If the powertrain contains an additional electric motor, the entire target torque is assigned to this motor (which corresponds to the strategy mode ElecDrive). If the powertrain model contains a disengaged clutch between combustion engine and electric motor, EngineStop opens this clutch.

To enter EngineStop, following conditions have to be fulfilled for a specific observation time:

- Combustion engine is on
- Combustion engine should be off for the target strategy mode

EngineStop is active until the combustion engine is shut down.

Parameter EngineStop

PowerTrain.Control.EngStop.tObsv = *value*

Observation time before entering the transition mode EngineStop. Default: 1.0s.

15.2.7 Battery management

Battery management controls the batteries' state of charge (SOC) by assigning generator torques to the integrated starter generator (ISG) and the electric motor and managing the energy transfer between the electric circuits of the power supply model. Its functionality is closely linked to the architecture of power supply, but its general intention is always to keep each batteries' SOC at its target value. In order to reach this goal, PTControl battery management contains control loops over the total power P_{tot} in the electric circuits and the battery's state of charge SOC. Its target state of charge SOC_{target} and its control factors I_{SOC} and I_P can be set by the user.

The following paragraphs describe IPGPowertrain PTControl battery management for different power supply architectures.

One electric circuit with ISG

The PTControl model like 'Micro' contains a battery management for power supply models with only one electric circuit and ISG. This battery management consists of one control loop:

- If the current strategy mode is EngineDrive, it assigns an additional (negative) generator torque $T_{ISGtarget}$ to the ISG in order to recharge the battery. At the same time, the combustion engine's target torque is increased so that the overall driving torque does not drop. Calculation of the target generator torque:

$$P_{target} = I_{SOC} \cdot \int (SOC - SOC_{target}) - I_P \cdot \int P_{tot_{LV}} \quad \text{(EQ 139)}$$

$$T_{ISGtarget} = \frac{P_{target}}{\omega_{ISG}} \quad \text{(EQ 140)}$$

The target generator torque is limited by the available surplus torque (difference between the target combustion engine's target torque and combustion engine's maximum torque).

Two electric circuits with ISG and/or electric motor

The PTControl models like 'Parallel P1' or 'Parallel P2' contains a battery management for power supply models with two electric circuit (low voltage and high voltage) and one ISG or electric motor or both. This battery management consists of two control loops:

- If the current strategy mode is EngineDrive, it assigns an additional (negative) generator torque to the ISG or to the electric motor in order to recharge battery A (battery on whose electric circuit the generator/motor is connected). At the same time, the combustion engine's target torque is increased so that the overall driving torque does not drop. Calculation of the target generator torque:

$$P_{target_A} = I_{SOC_A} \cdot \int (SOC_A - SOC_{target_A}) - I_{P_A} \cdot \int P_{tot_A} \quad \text{(EQ 141)}$$

$$T_{ISGtarget} = \frac{P_{target_A}}{\omega_{ISG}} \quad \text{(EQ 142)}$$

- If the SOC of battery A is bigger than its minimum admissible value, the target power transfer from circuit A to circuit B $P_{AtoBtarget}$ for the DC/DC is set to P_{target_B}:

$$P_{AtoBtarget} = P_{target_B} = I_{SOC_B} \cdot \int (SOC_B - SOC_{target_B}) - I_{P_B} \cdot \int P_{tot_B} \quad \text{(EQ 143)}$$

Parameter for Battery Management

PowerTrain.Control.BattMan.Active = *bool*

Add battery management to PTControl.

PowerTrain.Control.BattMan.LV.Pwr_Gen_max = *value*
PowerTrain.Control.BattMan.HV.Pwr_Gen_max = value

Maximum admissible generator power (absolute value) for battery management in low / high voltage circuit. Default: 2.0kW.

PowerTrain.Control.BattMan.LV.SOC_trg = *value*
PowerTrain.Control.BattMan.HV.SOC_trg = value

Target state of charge of low / high voltage battery for battery management. Default: 70%.

PowerTrain.Control.BattMan.LV.i_SOC = *value*
PowerTrain.Control.BattMan.HV.i_SOC = *value*

Control parameter (integral element) for low / high voltage battery's state of charge control. Default: 50.

PowerTrain.Control.BattMan.LV.i_Pwr = *value*
PowerTrain.Control.BattMan.HV.i_Pwr =*value*

Control parameter (integral element) for low / high voltage electric circuit's consumer control. Default: 1.0.

15.2.8 PTControl Models

The description of the PTControl models from the powertrain components library is described in the corresponding powertrain module (see section 15.7 'Powertrain Models').

PTControl model "User C-code / Simulink Plug-in / FMU"

The following parameters are required for the initialization of the output variables in the interface struct *tPTControlCfgIF*:

PowerTrain.Control.StartEngineWithSST = *bool*

Control element for switching operation states: 1 = switch operation states with start button, 0 = switch operation states with key only. Default: 0.

PowerTrain.Control.Velocity_max = *value*

Specifies the vehicle maximum possible velocity for instruments. Default: 260km/h.

Please note:

The following output variables have to be set to -99999 (= "not set") if the signals are not provided by the PTControl model. For Simulink Plug-ins the signals have to be set explicitly to -99999 else they will be set to 0.

If the output signals are set to a value different from -99999 (="not set"), they overwrite the target values calculated by the corresponding control unit (ECU, MCU, TCU). E.g. the target gear defined by TCU can be overwritten by PTControl. However, only one target value may be set at a time!

Output Variable	Unit	Description
EngineOut.FuelCutOff	bool	Flag if fuel is cut-off
EngineOut.Load	-	Engine target load
EngineOut.Trq_trg	Nm	Engine target torque
EngineOut.rotv_trg	rad/s	Engine target rotation speed
ISGOut.Load MotorOut<*m*>.Load	-	Integrated starter generator / motor *m* target load
ISGOut.Trq_trg MotorOut<*m*>.Trq_trg	Nm	Integrated starter generator / motor *m* target torque
ISGOut.rotv_trg MotorOut<*m*>.rotv_trg	rad/s	Integrated starter generator / motor *m* target rotation speed
ClutchOut.Pos	-	Clutch target position
ClutchOut.rotv_out_trg	rad/s	Clutch output shaft target rotation speed
ClutchOut.Trq_out_trg	Nm	Clutch output shaft target torque
GearBoxOut.GearNoTrg GearBoxM_Out<*gb*>.GearNoTrg GearBoxOut.GearNoTrg_dis GearBoxM_Out<*gb*>.GearNoTrg_dis	-	Gearbox target gear

Output Variable	Unit	Description
GearBoxOut.i_trg GearBoxM_Out<*gb*>.i_trg	bool	Gearbox target ratio
GearBoxOut.rotv_in_trg GearBoxM_Out<*gb*>.rotv_in_trg	rad/s	Gearbox input shaft target rotation speed
GearBoxOut.Trq_out_trg GearBoxM_Out<gb>.Trq_out_trg	Nm	Gearbox output shaft target torque
GearBoxOut.Clutch.Pos GearBoxM_Out<*gb*>.Clutch.Pos GearBoxOut.Clutch_dis.Pos GearBoxM_Out<*gb*>.Clutch_dis.Pos	-	Gearbox clutch target position
GearBoxOut.Clutch.rotv_out_trg GearBoxM_Out<*gb*>.Clutch.rotv_out_trg GearBoxOut.Clutch_dis.rotv_out_trg GearBoxM_Out<*gb*>.Clutch_dis.rotv_out_trg	rad/s	Gearbox clutch output shaft target rotation speed
GearBoxOut.Clutch.Trq_out_trg GearBoxM_Out<*gb*>.Clutch.Trq_out_trg GearBoxOut.Clutch_dis.Trq_out_trg GearBoxM_Out<*gb*>.Clutch_dis.Trq_out_trg	Nm	Gearbox clutch output shaft target torque
PwrSupplyOut.Pwr_HV1toLV_trg	W	Target transferred electric power from high voltage 1 electric circuit to low voltage electric circuit

15.3 Control Units

In the structure of CarMaker powertrain models, the control units bridge the gap between the PTControl model and the electro mechanic powertrain component models. Their main tasks are

- Regulation of the electro mechanic components in order to establish the target values (e.g. torque, speed) that are provided by PTControl
- Evaluate the current state of the electro mechanic components

CarMaker powertrain model contains up to four control units. If several instances of one component (e.g. several electric motors) exist, the corresponding control unit model (e.g. MCU) pilots all of them. This means, the powertrain does not have several control unit models of the same kind.

Control Unit	Abbrev.	Controlled components
Engine control unit	ECU	Combustion engine
Motor control unit	MCU	Internal starter generator, all electric motors
Transmission control unit	TCU	All gearboxes, all clutches inside gearbox models
Battery control unit	BCU	Power supply with all batteries inside

15.3.1 Engine control unit (ECU)

The ECU model controls the powertrain's combustion engine. Its main tasks are:

- Observation of the engine's state (on/off)
- Precontrol and control of the engine's load in order to establish the target torque, target rotation speed or target load that is provided from PTControl
- Idle speed control
- Fuel cut-off detection
- Determination of engine's full load torque, drag torque and torque with optimum consumption at the current rotation speed

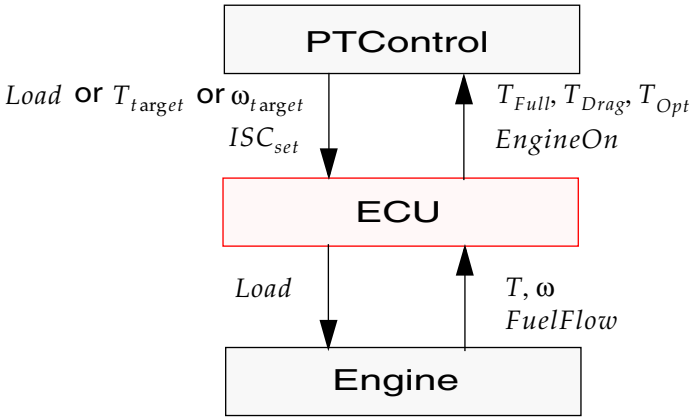


Figure 15.23: ECU model

PowerTrain.ECU.Kind = *KindStr VersionId*

Selection of the engine control unit subsystem to use. Following submodels are provided:

ModelName	KindStr	Description
Basic	Basic	Standard IPGPowertrain ECU model

```
PowerTrain.ECU.Kind = Basic 1
```

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tPTEngineCU_CfgIF*:

Input Variable	Unit	Description
rotv_off	rad/s	Engine off rotation speed
rotv_max	rad/s	Engine maximum rotation speed
rotv_idle	rad/s	Engine idle speed
rotv_opt	rad/s	Engine optim. speed with minimum consumption
TrqFull	Nm	1D-Lookup table of maximum engine full torque
TrqDrag	Nm	1D-Lookup table of maximum engine drag torque
TrqOpt	Nm	1D-Lookup table of engine torque with optimum consumption as function of engine speed

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tPTEngineCU_IF*:

Input Variable	Unit	Description
Ignition	bool	Vehicle ignition
set_ISC	bool	Idle speed controller activated
FuelCutOff	bool	Fuel is cut-off forced by PTControl
Load	-	Engine target load forced by PTControl
Trq_trg	Nm	Engine target torque
Trq	Nm	Current engine torque
rotv_trg	rad/s	Engine target rotation speed
rotv	rad/s	Current engine rotation speed
rot	rad	Current engine rotation angle
FuelFlow	l/s	Current engine fuel flow

Output Variable	Unit	Description
Status	-	ECU status
Engine_on	bool	Flag if engine is on
FuelCutOff	bool	Fuel is cut-off
Load	-	Load/throttle signal for engine
TrqDrag	Nm	Engine drag torque at current rotation speed

Output Variable	Unit	Description
TrqFull	Nm	Engine full torque at current rotation speed
TrqOpt	Nm	Engine torque with optimum consumption at current rotation speed

ECU Model "Basic"

The ECU model *Basic* sets the engine's load and determines its full load torque, its drag torque and its optimum torque at current engine rotation speed.

The engine's load is calculated based on either a target torque and its difference to the current engine torque or a target rotation speed and its difference to the current rotation speed or a target load. The kind of engine control (torque, rotation speed or load) depends on the target value given by PTControl. The ECU model *Basic* can execute one kind of engine control at a time. A target value of -99999 is interpreted as "not set" and thus ignored by this ECU model.

The idle speed control is independent of the engine's load calculation and becomes active, if the engine's rotation speed drops below its turn off rotation speed (Infofile parameter *PowerTrain.ECU.ISCtrl.rotvOff*). It increases the engine load in order to keep its rotation speed above its idle speed.

Parameter Basic

PowerTrain.ECU.ISCtrl.Active = *bool*

Activate the idle speed controller. Default: 1.

PowerTrain.ECU.ISCtrl.p = *value*

Specifies the proportional value of the PI-idle speed controller. Default: 0.1

PowerTrain.ECU.ISCtrl.i = *value*

Specifies the integral value of the PI-idle speed controller. Default: 50.0

PowerTrain.ECU.ISCtrl.rotvOff = *value*

Turn off engine speed of the idle speed controller. Default: engine idle speed [rpm] + 200rpm.

PowerTrain.ECU.ISCtrl.LoadMax = *value*

Specifies the maximal load value of the idle speed controller. Default: 1.0.

PowerTrain.ECU.nFuelCutOff = *value*

Specifies the engine speed limit above which the fuel is cut-off.
Default: engine idle speed [rpm] * 2.

PowerTrain.ECU.TrqCtrl.p = *value*

Specifies the proportional value of the PI-torque controller. Default: 0.002.

PowerTrain.ECU.TrqCtrl.i = *value*

Specifies the integral value of the PI-torque controller. Default: 5.0.

PowerTrain.ECU.RotvCtrl.p = *value*

Specifies the proportional value of the PI-rotation speed controller. Default: 0.1.

PowerTrain.ECU.RotvCtrl.i = *value*

Specifies the integral value of the PI-rotation speed controller. Default: 50.0.

15.3.2 Motor control unit (MCU)

In CarMaker, the powertrain model may contain one integrated starter generator (ISG) model and up to eight instances of electric motor models (called Motor, Motor1, Motor2,...), but it only contains one single MCU model. This MCU model controls all electric motors including the integrated starter generator. Its main tasks are (for the integrated starter motor and each motor):

- Determination of the maximum motor torque and the maximum generator torque at the current motor rotation speed
- Precontrol and control of the motor's load in order to establish the target torque, target rotation speed or target load that is provided from PTControl
- Limit the motors rotation speed

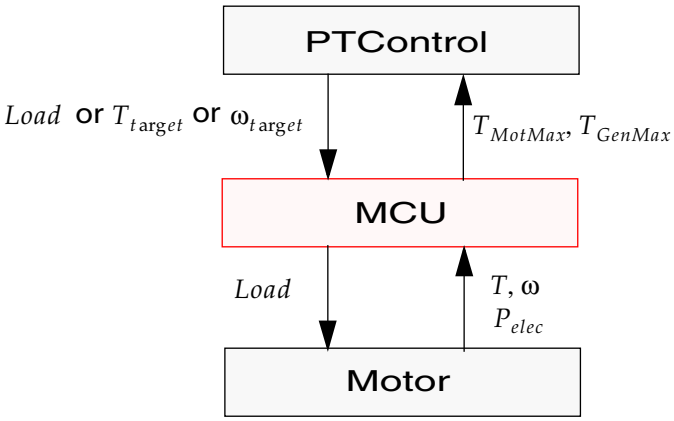


Figure 15.24: MCU model (for one electric motor)

The MCU calculates torques and rotation speeds at the reference point that is located on the driven shaft.

PowerTrain.MCU.Kind = *KindStr VersionId*

Selection of the motor control unit subsystem to use. The powertrain library provides the following submodels:

ModelName	KindStr	Description
Basic	Basic	Standard IPGPowertrain MCU model

```
PowerTrain.MCU.Kind = Basic 1
```

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tPTMotorCU_CfgIF*:

Input Variable	Unit	Description
nMotor		Number of motors (without ISG)
ISG.Ratio Motor<*m*>.Ratio		Integrated starter generator / motor *m* ratio between motor shaft and driven shaft

Input Variable	Unit	Description
ISG.rotv_Mot_max Motor<*m*>.rotv_Mot_max	rad/s	Integrated starter generator / motor *m* maximum motor rotation speed
ISG.rotv_Gen_max Motor<*m*>.rotv_Gen_max	rad/s	Integrated starter generator / motor *m* maximum generator rotation speed
ISG.TrqMot_max Motor<*m*>.TrqMot_max	Nm	1D-Lookup table of integrated starter generator / motor *m* maximum motor torque [Nm]
ISG.TrqGen_max Motor<*m*>.TrqGen_max	Nm	1D-Lookup table of integrated starter generator / motor *m* maximum generator torque [Nm]

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tPTMotorCU_IF*:

Input Variable	Unit	Description
Ignition	bool	Vehicle ignition
ISGOut.Load MotorOut<*m*>.Load	-	Integrated starter generator / motor *m* target load forced by PTControl
ISGIn.Trq_trg MotorIn<*m*>.Trq_trg	Nm	Integrated starter generator / motor *m* target torque
ISGIn.Trq MotorIn<*m*>.Trq	Nm	Current integrated starter generator / motor *m* torque
ISGIn.rotv_trg MotorIn<*m*>.rotv_trg	rad/s	Integrated starter generator / motor *m* target rotation speed
ISGIn.rotv MotorIn<*m*>.rotv	rad/s	Current integrated starter generator / motor *m* rotation speed
ISGIn.PwrElec MotorIn<*m*>.PwrElec	W	Current electric power of integrated starter generator / motor *m*

Output Variable	Unit	Description
Status	-	MCU status
ISGOut.Load MotorOut<*m*>.Load	-	Load signal for integrated starter generator / motor *m*
ISGOut.TrqMot_max MotorOut<*m*>.TrqMot_max	Nm	Integrated starter generator / motor *m* maximum motor torque at current rotation speed
ISGOut.TrqGen_max MotorOut<*m*>.TrqGen_max	Nm	Integrated starter generator / motor *m* maximum generator torque at current rotation speed

MCU Model "Basic"

The MCU model *Basic* sets each motor's load and determines its maximum motor and generator torques at the current rotation speed. Additionally, it limits the motors' rotation speeds by setting the load to zero, if the rotation speed exceeds its maximum.

Each motor's load is calculated based on either a target torque and its difference to the current motor torque or a target rotation speed and its difference to the current rotation speed or a target load. The kind of motor control (torque, rotation speed or load) depends on the target value given by PTControl. The MCU model *Basic* can execute one kind of motor control for each motor at a time. A target value of -99999 is interpreted as "not set" and thus ignored by this MCU model.

In the model *Basic* the maximum motor/generator torque on the output interface corresponds to the maximum possible mechanical torque without considering the electrical power charge/discharge limitation by the battery. This limitation is considered in the corresponding PTControl model.

Parameter Basic

PowerTrain.MCU.<*m*>.TrqCtrl.p = *value*

Specifies the proportional value of the PI-torque controller for integrated starter generator / electric motor *m* (<*m*> := MotorISG, Motor, Motor1, Motor2,...). Default: 0.002.

PowerTrain.MCU.<*m*>.TrqCtrl.i = *value*

Specifies the integral value of the PI-torque controller for integrated starter generator / electric motor *m* (<*m*> := MotorISG, Motor, Motor1, Motor2,...). Default: 5.0.

PowerTrain.MCU.<*m*>.RotvCtrl.p = *value*

Specifies the proportional value of the PI-rotation speed controller for integrated starter generator / electric motor *m* (<*m*> := MotorISG, Motor, Motor1, Motor2, ...). Default: 0.1.

PowerTrain.MCU.<*m*>.RotvCtrl.i = *value*

Specifies the integral value of the PI-rotation speed controller for integrated starter generator / electric motor *m* (<*m*> := MotorISG, Motor, Motor1, Motor2,...). Default: 50.0.

15.3.3 Transmission control unit (TCU)

In CarMaker, the powertrain model may contain one gearbox model for each combustion engine (called GearBox) and electric motor (called GearBoxM, GearBoxM1,...), but it only contains one single TCU model. This TCU model controls all automatic gearboxes including their clutches.

Its main tasks are (for each automatic gearbox):

- Determination of a target gear according to gear selector position or gear ratio in case of a CVT gearbox
- Control of the clutch position (friction clutch or hydraulic converter lockup clutch)

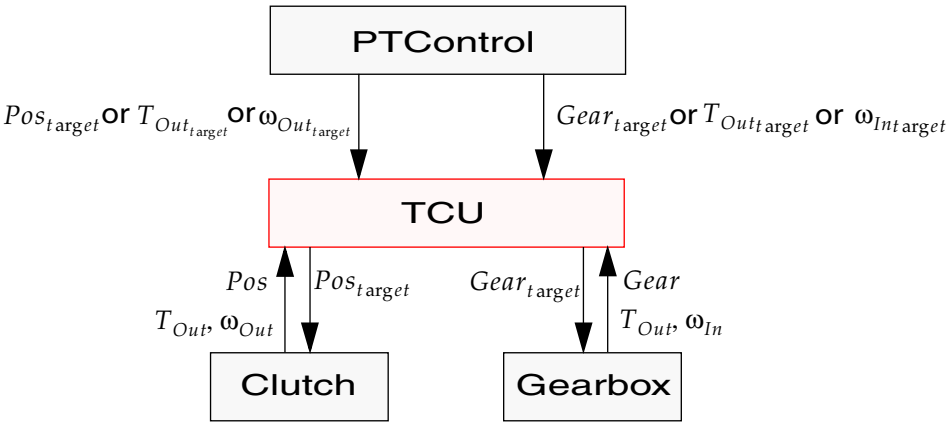


Figure 15.25: TCU model (for one clutch and gearbox)

The selection of the gear ratio with the manual transmission model is done by user requirements or the driver model (IPGDriver).

Using an automatic transmission, it is possible to impose a target gear number (Manumatic) by user or by the driver model (IPGDriver). If the selector control position *DM.SelectorCtrl* equals *SelectorCtrl_M* (:=2), the manual mode is activated and the target gear number can be set in quantity *DM.GearNo*.

- If the difference between the current gear number and the user defined gear number is more than one gear position, the automatic transmission changes gear step by step.
- The manual gear shifting mode is only possible if the TCU supports this feature (*AutoWithMan*=1 in the initialization interface struct).

PowerTrain.TCU.Kind = *KindStr VersionId*

Selection of the transmission control unit subsystem to use. The powertrain library provides the following submodels:

ModelName	KindStr	Description
Automatic	Automatic	Standard IPGPowertrain transmission control unit for automatic gearbox without internal clutch
Automatic with Converter	Auto_Conv	Standard IPGPowertrain transmission control unit for automatic gearbox with internal hydraulic converter clutch
Automated Manual Transmission	Auto_AMT	Standard IPGPowertrain transmission control unit for automatic gearbox with automated manual transmission (with internal friction clutch)
Automatic with Converter and Automated Manual Transmission	Auto_Conv _AMT	Standard IPGPowertrain transmission control unit for automatic gearbox with internal hydraulic converter clutch for the main gearbox and with automated manual transmission for the electrical motor gearbox
Continuously Variable Transmission	CVT	Standard IPGPowertrain transmission control unit for continuously variable transmission
Dual Clutch Transmission	DCT	Standard IPGPowertrain transmission control unit for automatic gearbox with dual clutch transmission

Example `PowerTrain.TCU.Kind = Automatic 1`

Parameter Gearbox

The gearbox models that are provided by the powertrain components library have some parameters in common.

Parameters for the gear shifting logic

Following parameters for the gear shifting logic are required for all IPGPowertrain TCU models (except for *Continuously Variable Transmission*).

Parameter Gear Shifting Logic

PowerTrain.TCU.<i>.Active = *bool*

Activation of the gearbox <i> control by the TCU model (<i> := GB, GB_M, GB_M1, ...).

PowerTrain.TCU.<i>.RotvRef = *RotvRefStr*

Selection of the observed reference speed for the shifting logic for gearbox i (<i> := ShiftCtrl, ShiftCtrlM, ShiftCtrlM1, ...).
Possible values are: Engine, GearBox_In, GearBox_Out. Default: GearBox_In.

PowerTrain.TCU.<i>.Delay = *value*

State the minimal time between two shifting operations for gearbox i (<i> := ShiftCtrl, ShiftCtrlM, ShiftCtrlM1, ...). Default: 0.5s.

PowerTrain.TCU.<i>.Kind = *KindStr*

Selection of the gearbox control unit's shifting logic for gearbox i (<i> := ShiftCtrl, ShiftCtrlM, ShiftCtrlM1, ...).

The gearbox components library provides the following shifting logic models:

ModelName	KindStr	Description
1D Look-Up Table	Linear	Gear shifting logic with minimal/maximal shifting limits for each gear.
2D Look-Up Table	Linear2D	Gear shifting logic with minimal/maximal shifting limits for each gear and as function of gas pedal.

Example `PowerTrain.TCU.ShiftCtrl.Kind = Linear`

PowerTrain.TCU.<i>.nShift: *ShiftLimitsList*

For shifting logic model 1D Look-Up Table:
State the minimal and maximal shifting limits for each gear number for gearbox i (<i> := ShiftCtrl, ShiftCtrlM, ShiftCtrlM1, ...). Values of last gear element in the list are valid for all remaining gears.

Syntax Infofile table mapping with 3 columns
<gear number> <nMin> <nMax>

Example
```
PowerTrain.TCU.ShiftCtrl.nShift:
1   1000  3000
2   1500  2500
...
```

PowerTrain.TCU.<i>.nShift: *ShiftLimitsList*

For shifting logic model 2D Look-Up Table:

State the minimal and maximal shifting limits for each gear number as function of gas pedal for gearbox i (<i> := ShiftCtrl, ShiftCtrlM, ShiftCtrlM1, ...). Values of last gear element in the list are valid for all remaining gears.

Syntax Infofile table mapping with 4columns
<gear number> <Gas> <nMin> <nMax>

Example
```
PowerTrain.TCU.ShiftCtrl.nShift:
1   0.0   1000   2000
1   0.5   1000   2500
1   1.0   1000   3000
2   0.0   1500   2500
...
```

Parameter
Manual Shifting

PowerTrain.TCU.AutoWithMan = *bool*

Specifies if the TCU supports manual gear selection (Manumatic); only for main gearbox (gearbox in combination with combustion engine). Default: 1.

PowerTrain.TCU.<i>.Man.SpeedRange = *nMin nMax*

State the minimal and maximal engine or motor speed range while the manual shifting process for gearbox i (<i> := ShiftCtrl, ShiftCtrlM, ShiftCtrlM1, ...). If the engine speed reaches the limits and the driver doesn't react, the gearbox shifts itself up or down.
Default: 800rpm / 7000rpm.

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tPTTransmCU_CfgIF*:

Input Variable	Description
nGearBoxM	Number of gearboxes (without gearbox for engine)
ClKind	Kind of the clutch after engine
GearBox.GBKind GearBoxM<*gb*>.GBKind	Gearbox kind
GearBox.ClKind GearBoxM<*gb*>.ClKind	Gearbox internal clutch kind
GearBox.nFGears GearBoxM<*gb*>.nFGears	Gearbox number of forward gears
GearBox.iFGear[*gear*] GearBoxM<*gb*>.iFGear[*gear*]	Gearbox ratios of forward gears
GearBox.nBGears GearBoxM<*gb*>.nBGears	Gearbox number of backward gears
GearBox.iBGear[*gear*] GearBoxM<*gb*>.iBGear[*gear*]	Gearbox ratios of backward gears

Output Variable	Description
AutoWithMan	Autom. gearbox with manual gear selection (Manumatic)

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tPTTransmCU_IF*:

Input Variable	Unit	Description
Ignition	bool	Vehicle ignition
Load	-	Engine throttle
SelectorCtrl	-	Gearbox selector control
ClutchOut.Pos GearBoxOut.Clutch.Pos GearBoxOut.Clutch_dis.Pos GearBoxM_Out<*gb*>.Clutch.Pos GearBoxM_Out<*gb*>.Clutch_dis.Pos	-	Clutch target position forced by PTControl
ClutchIn.rotv_in GearBoxIn.Clutch.rotv_in GearBoxIn.Clutch_dis.rotv_in GearBoxM_In<*gb*>.Clutch.rotv_in GearBoxM_In<gb>.Clutch_dis.rotv_in	rad/s	Clutch input shaft rotation speed
ClutchIn.rotv_out GearBoxIn.Clutch.rotv_out GearBoxIn.Clutch_dis.rotv_out GearBoxM_In<*gb*>.Clutch.rotv_out GearBoxM_In<*gb*>.Clutch_dis.rotv_out	rad/s	Clutch output shaft rotation speed
ClutchIn.rotv_out_trg GearBoxIn.Clutch.rotv_out_trg GearBoxIn.Clutch_dis.rotv_out_trg GearBoxM_In<*gb*>.Clutch.rotv_out_trg GearBoxM_In<*gb*>.Clutch_dis.rotv_out_trg	rad/s	Clutch output shaft target rotation speed
ClutchIn.Trq_in GearBoxIn.Clutch.Trq_in GearBoxIn.Clutch_dis.Trq_in GearBoxM_In<*gb*>.Clutch.Trq_in GearBoxM_In<*gb*>.Clutch_dis.Trq_in	Nm	Clutch input shaft torque
ClutchIn.Trq_out GearBoxIn.Clutch.Trq_out GearBoxIn.Clutch_dis.Trq_out GearBoxM_In<*gb*>.Clutch.Trq_out GearBoxM_In<*gb*>.Clutch_dis.Trq_out	Nm	Clutch output shaft torque
ClutchIn.Trq_out_trg GearBoxIn.Clutch.Trq_out_trg GearBoxIn.Clutch_dis.Trq_out_trg GearBoxM_In<*gb*>.Clutch.Trq_out_trg GearBoxM_In<*gb*>.Clutch_dis.Trq_out_trg	Nm	Clutch output shaft target torque
ClutchIn.i_TrqIn2Out GearBoxIn.Clutch.i_TrqIn2Out GearBoxIn.Clutch_dis.i_TrqIn2Out GearBoxM_In<*gb*>.Clutch.i_TrqIn2Out GearBoxM_In<*gb*>.Clutch_dis.i_TrqIn2Out	-	Clutch ratio input to output torque (estimated)
GearBoxIn.GearNo GearBoxM_In<*gb*>.GearNo	-	Gearbox current gear
GearBoxIn.GearNo_dis GearBoxM_In<*gb*>.GearNo_dis	-	Gearbox current gear for a optional second disengaged shaft

Input Variable	Unit	Description
GearBoxOut.GearNoTrg GearBoxM_Out<*gb*>.GearNoTrg	-	Gearbox target gear forced by PTControl
GearBoxOut.GearNoTrg_dis GearBoxM_Out<*gb*>.GearNoTrg_dis	-	Gearbox target gear forced by PTControl for optional second disengaged shaft
GearBoxIn.i GearBoxM_In<*gb*>.i	-	Gearbox current ratio
GearBoxIn.rotv_in GearBoxM_In<*gb*>.rotv_in	rad/s	Gearbox input shaft rotation speed
GearBoxIn.rotv_in_trg GearBoxM_In<*gb*>.rotv_in_trg	rad/s	Gearbox input shaft target rotation speed
GearBoxIn.rotv_out GearBoxM_In<*gb*>.rotv_out	rad/s	Gearbox output shaft rotation speed
GearBoxIn.Trq_in GearBoxM_In<*gb*>.Trq_in	Nm	Gearbox input shaft torque
GearBoxIn.Trq_out GearBoxM_In<*gb*>.Trq_out	Nm	Gearbox output shaft torque
GearBoxIn.Trq_out_trg GearBoxM_In<*gb*>.Trq_out_trg	Nm	Gearbox output shaft target torque

Output Variable	Unit	Description
Status	-	TCU status
ClutchOut.Pos GearBoxOut.Clutch.Pos GearBoxOut.Clutch_dis.Pos GearBoxM_Out<*gb*>.Clutch.Pos GearBoxM_Out<*gb*>.Clutch_dis.Pos	-	Clutch target position
GearBoxOut.GearNoTrg GearBoxM_Out<*gb*>.GearNoTrg	-	Gearbox target gear
GearBoxOut.GearNoTrg_dis GearBoxM_Out<*gb*>.GearNoTrg_dis	-	Gearbox target gear for optional second disengaged shaft
GearBoxOut.set_ParkBrake GearBoxM_Out<*gb*>.set_ParkBrake	bool	Set gearbox park brake
GearBoxOut.i_trg GearBoxM_Out<*gb*>.i_trg	-	Gearbox target ratio
GearBoxOut.Trq_DriveSrc_trg GearBoxM_Out<*gb*>.TrqDriveSrc_trg	Nm	Optional drive source target torque from TCU to PTControl (e.g. while shifting)

TCU Model "Automatic"

The TCU model *Automatic* is designed for an automatic gearbox model (gearbox behind the engine) with an external hydraulic converter model (that is NOT part of the gearbox model). This topology can be found e.g. in the powertrain model Generic (see section 15.7.4 'Powertrain model "Generic"').

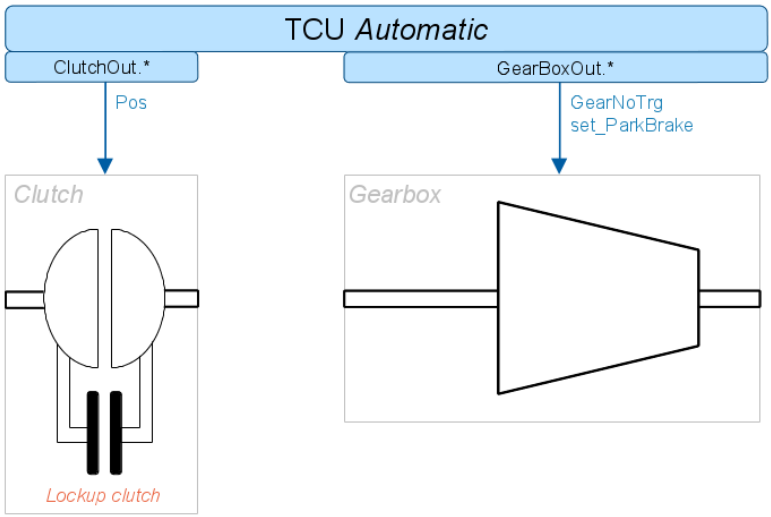


Figure 15.26: TCU model "Automatic"

This TCU model sets a target gear number for the gearbox based on the gear selector's position and the powertrain's rotation speed. Furthermore, it opens the hydraulic converter lockup clutch during gear shifting. Target gear and clutch position that are input into this TCU model from PTControl, driver model or external file obtain priority.

TCU Model "Automatic with Converter"

The TCU model *Automatic with Converter* is designed for an automatic gearbox model (gearbox behind the engine) with an internal hydraulic converter model (that is part of the gearbox model). This topology can be found e.g. in the powertrain model Parallel P2 (see section 'Configuration "Parallel P2"').

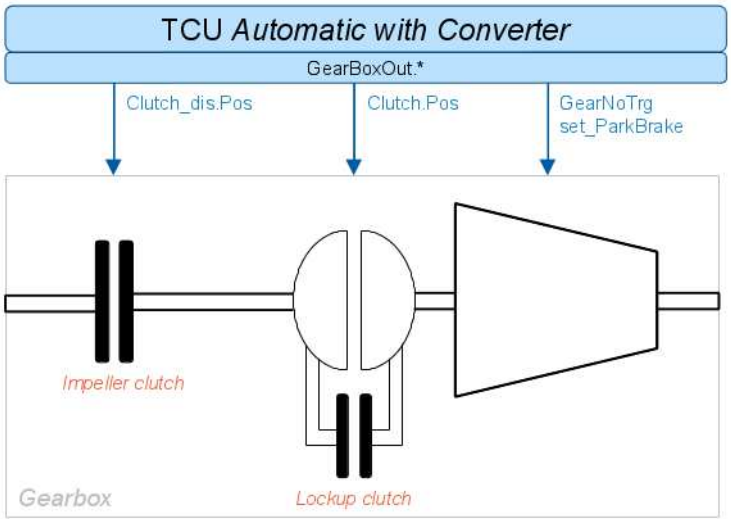


Figure 15.27: TCU model "Automatic with Converter"

This TCU model sets a target gear number for the gearbox based on the gear selector's position and the powertrain's rotation speed. Furthermore, it opens the hydraulic converter lockup clutch during gear shifting. Target gear and clutch position that are input into this TCU model from PTControl, driver model or external file obtain priority.

Parameters for the converter lockup clutch

Following parameters for the converter lockup clutch are required for the TCU models *Automatic* and *Automatic with Converter*.

Parameter Lockup Clutch

PowerTrain.TCU.LockUp.Kind = *KindStr*

Specifies the kind of the lock-up activation:

ModelName	KindStr	Description
DVA	DVA	Lock-up activated via DVA on “PT.TCU.LockUp”
2D Look-Up Table	Linear2D	Lock-up activation depending on logic with lock/unlock limits for each gear and as function of load. 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 falls below the unlock speed limit.

PowerTrain.TCU.LockUp.nLock: *Lock/UnlockLimitsList*

State the lock and unlock speed limits for each gear number as function of engine load. Values of last gear element in the list are valid for all remaining gears.

Syntax

Infofile table mapping with 4columns
<gear number> <load> <nUnlock> <nLock>

Example

```
PowerTrain.TCU.LockUp.nLock:
1   0.0  2000  3000
1   0.5  2500  3500
1   1.0  3000  4000
2   0.01  800  2800
...
```

PowerTrain.TCU.LockUp.RotvRef = *RotvRefStr*

Selection of the observed reference speed for the lock-up logic. Possible values are: Engine, GearBox_In, GearBox_Out. Default: GearBox_In.

PowerTrain.TCU.LockUp.Delay = *value*

State the minimal time between lock and unlock operation. Default: 0.5 s.

PowerTrain.TCU.LockUp.nFit = *value*

Specifies the time to close the lockup clutch (synchronization time). Default: 200ms.

TCU Model "Automated Manual Transmission"

The TCU model *Automated Manual Transmission* is designed for the control of the engine gearbox and for the control of all electric motor gearboxes, if they all are of type automated manual transmission with an internal friction clutch.

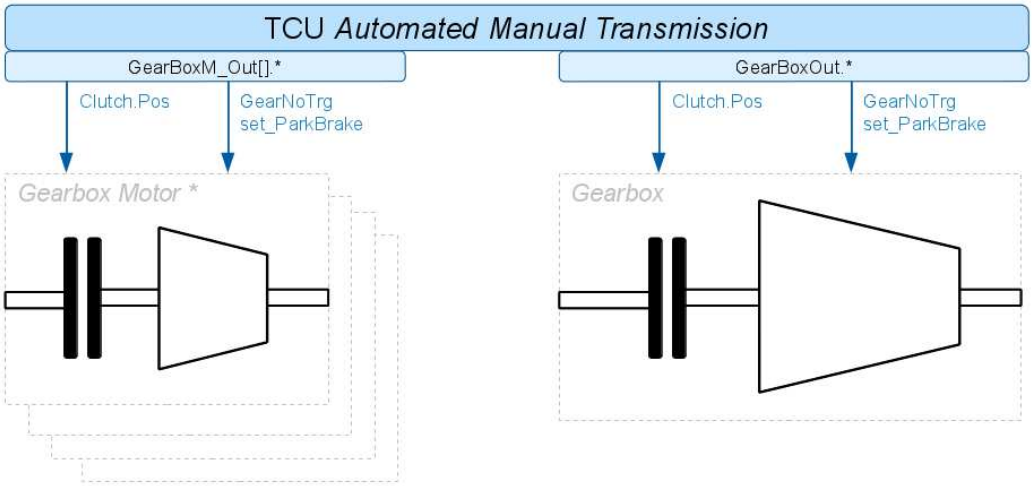


Figure 15.28: TCU model "Automated Manual Transmission"

This TCU model sets a target gear number for each gearbox based on the gear selector's position and the powertrain's rotation speed. Target gear and clutch position that are input into this TCU model from PTControl, driver model or external file obtain priority.

TCU Model "Automatic with Converter and Automated Manual Transmission"

The TCU model *Automatic with Converter and Automated Manual Transmission* is designed for the control of the engine gearbox using an internal hydraulic converter model and for the control of all electric motor gearboxes using a gearbox of type automated manual transmission with an internal friction clutch.

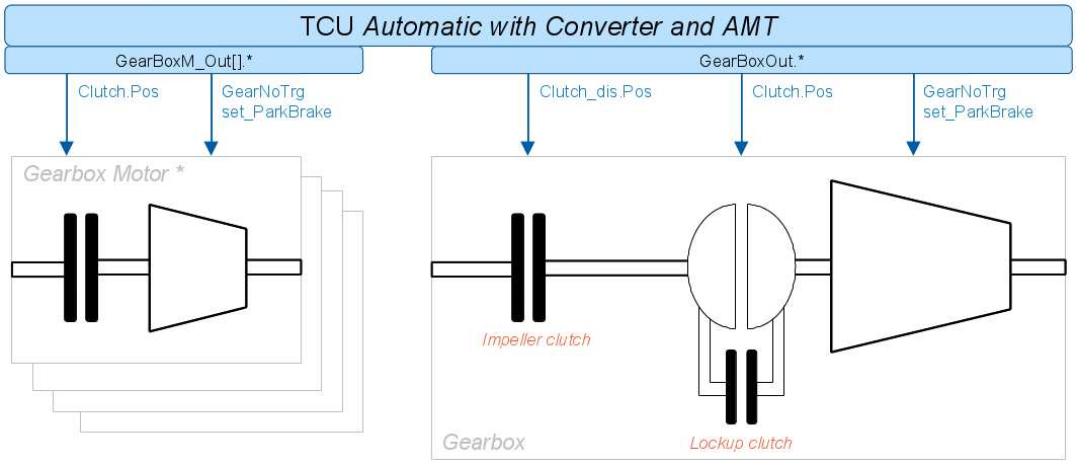


Figure 15.29: TCU model "Automatic with Converter and AMT"

This TCU model sets a target gear number for the gearbox based on the gear selector's position and the powertrain's rotation speed. Furthermore, it opens the hydraulic converter lockup clutch during gear shifting. Target gear and clutch position that are input into this TCU model from PTControl, driver model or external file obtain priority.

Parameter using Converter Clutch

PowerTrain.TCU.Trq_in_off_active = *bool*

Optional, only for the TCU models *Automatic* and *Automatic with Converter*.
Specifies if the TCU uses the interface signal *Trq_DriveSrc_trg* to require a zero drive source target torque to the main gearbox during gear shifting. Default: 0.

TCU Model "Continuously Variable Transmission"

The TCU model *Continuously Variable Transmission* (CVT) is designed for an automatic gearbox model of a continuously variable transmission with an external friction clutch model. It can only be used inside Conventional powertrain models.

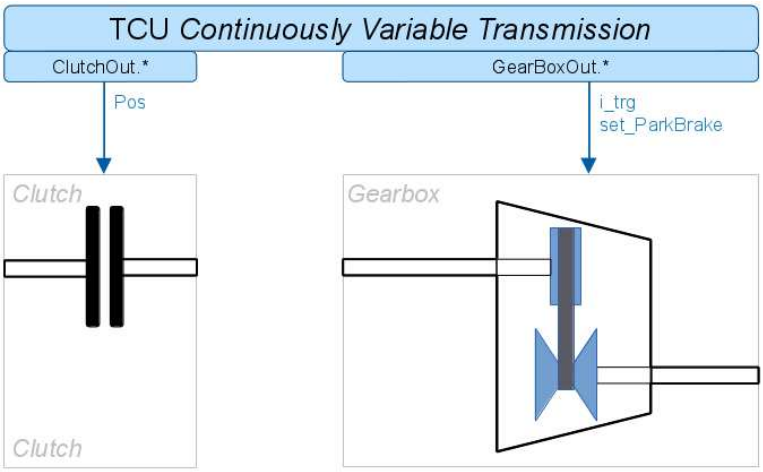


Figure 15.30: TCU model "Continuously Variable Transmission"

This TCU model sets a target transmission ratio for the gearbox based on the used controller mode (see below). Furthermore, it operates the friction clutch during start up and stand still. Target transmission ratio and clutch position that are input into this TCU model from PTControl, driver model or external file obtain priority.

There are two possible control modes (*speed envelope* and *single track*) to calculate the target transmission ratio.

Parameter Continuously Variable Transmission

PowerTrain.TCU.ControlMode = *ModeStr*

State the control mode for the calculation of the transmission ratio. Possible values are: SpeedEnv, SingleTrack.

Speed Envelope Strategy

The *speed envelope* strategy uses a two dimensional characteristic map to set the desired engine speed based on the accelerator pedal position and the vehicle velocity. The desired engine speed is then used to calculated the target transmission ratio.

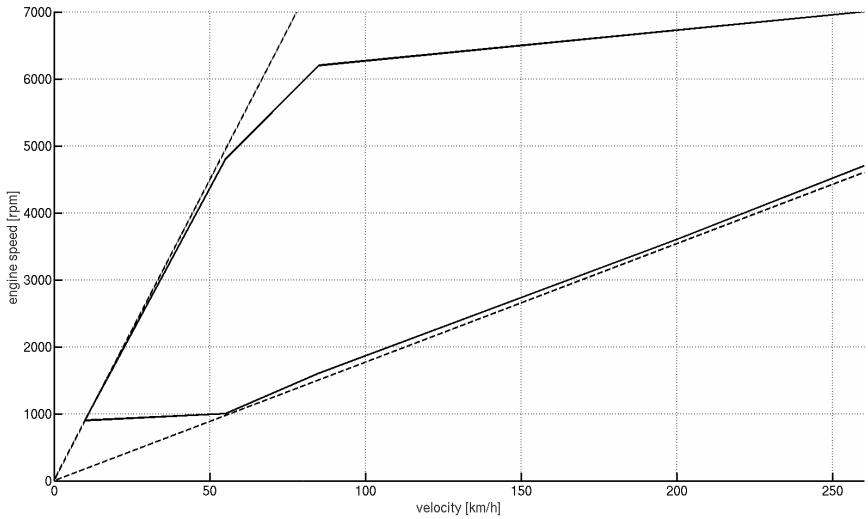


Figure 15.31: Speed envelope map

PowerTrain.TCU.SpeedEnvelope: *Table*

Two dimensional characteristic mapping for calculation of the transmission ratio. The map specifies the engine speed depending on the throttle position and the vehicle velocity. Please make sure to always adjust the map to the selected engine and the total transmission including differential gear and tires.

Syntax Infofile table mapping with 3 columns
<throttle position> [%] <vhcl. velocity> [km/h] <engine speed> [rpm]

Example

```
PowerTrain.GearBox.SpeedEnvelope:
0    10    900
0    55   1000
0    85   1600
0   200   3620
...
1   200   6000
1   260    6850
```

Single Track Strategy

The *single track* strategy uses the fuel consumption map of the engine model. The strategy also intervenes into the engine control by sending the desired engine torque to PTControl. From the consumption map a optimum operation line is created considering the minimum fuel consumption for a given engine speed. If the created optimum operation line is not monotonously increasing, it is adjusted accordingly. Additionally, a minimum target engine speed can be set. The strategy does not select a desired engine rotation speed that is smaller than this value. Due to this functionality, the engine is operated in a state where the consumption is minimal even if the demanded power is quite low.

The desired engine speed is calculated with the optimum operation line and the power demand from the accelerator pedal position. The target transmission ratio as well as the engine target torque will be calculated from the desired engine speed. There is no engine intervention during deceleration.

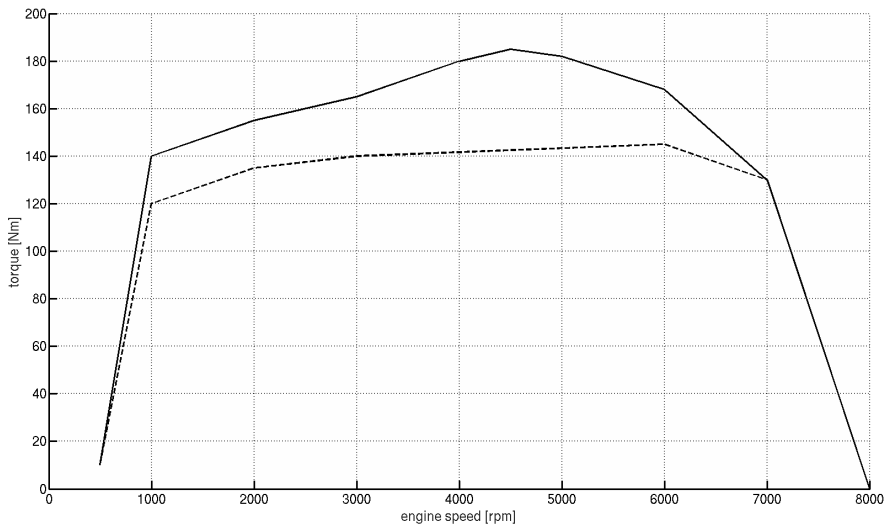


Figure 15.32: Single track optimal operation line (OOP)

PowerTrain.TCU.rotv_min = *Value*

State the minimum engine speed the control mode is allowed to select.
Default: 1200
Default: engine opt speed [rpm].

TCU Model "Dual Clutch Transmission"

The TCU model *Dual Clutch Transmission* is designed for an automatic gearbox model of a dual clutch transmission with the dual friction clutch being part of the gearbox model. It can be used for *Conventional* and hybrid *Parallel* powertrain.

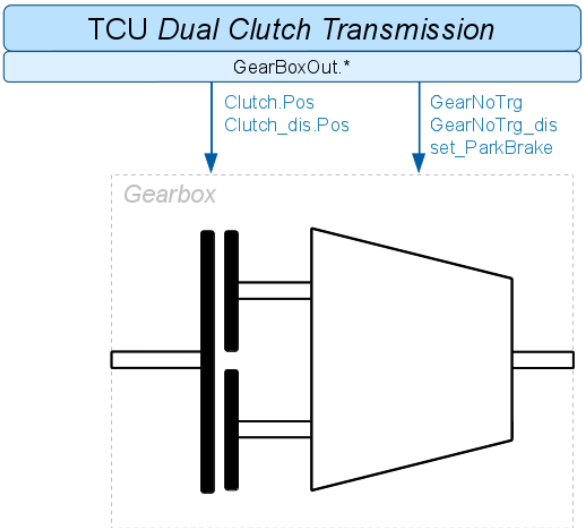


Figure 15.33: TCU model "Dual Clutch Transmission"

This TCU model sets a target gear number for the disengaged shaft based on the gear selector's position and the powertrain's rotation speed. Then it starts the gear shifting to the preselected gear on the disengaged shaft. Furthermore, it opens and closes both friction clutches during gear shifting to ensure a smooth transition. Target gear and clutch position that are input into this TCU model from PTControl, driver model or external file obtain priority.

The gear shifting is split into two phases: The torque phase, where the engine torque is transferred from the offgoing clutch to the incoming clutch and the inertia phase, where the rotation speed is synchronized. The order of these phases depends on whether it is a power-on upshift, power-off downshift or a power-off upshift, power-on downshift.

There are two different modes to control the gear shifting. The *Basic* mode will not intervene in the engine control, whereas the *Advanced* mode will control the engine speed to improve the shifting quality.

Parameter Dual Clutch Transmission

PowerTrain.TCU.ClutchCtrl.ControlMode = *ModeStr*

State the control mode for the gear shifting. Possible values are: Basic, Advanced.
Default: Basic

TCU model "User C-code / Simulink Plug-in / FMU"

Following parameters are required for the initialization of the output variables in the interface struct *tPTTransmCU_CfgIF*:

PowerTrain.TCU.AutoWithMan = *bool*

Specifies if the TCU supports manual gear selection (Manumatic); only for main gearbox (gearbox in combination with combustion engine). Default: 1

Please note:
The following output variables have to be set to -99999 (= "not set") if the signals are not provided by the TCU model. For Simulink Plug-ins the signals has to be set explicitly -99999 else they will be set to 0.

Output Variable	Unit	Description
GearBoxOut.Trq_DriveSrc_trg GearBoxM_Out<*gb*>.TrqDriveSrc_trg	Nm	Optional drive source target torque from TCU to PTControl (e.g. while shifting)

15.3.4 Battery control unit (BCU)

In CarMaker, the powertrain model may contain a low voltage and a high voltage battery, but it only contains one single BCU model. This BCU model controls the power supply and all battery models. Its main tasks are:

- Calculation of the state of charge and state of health of the batteries
- Transfer of signals between PTControl model and power supply model

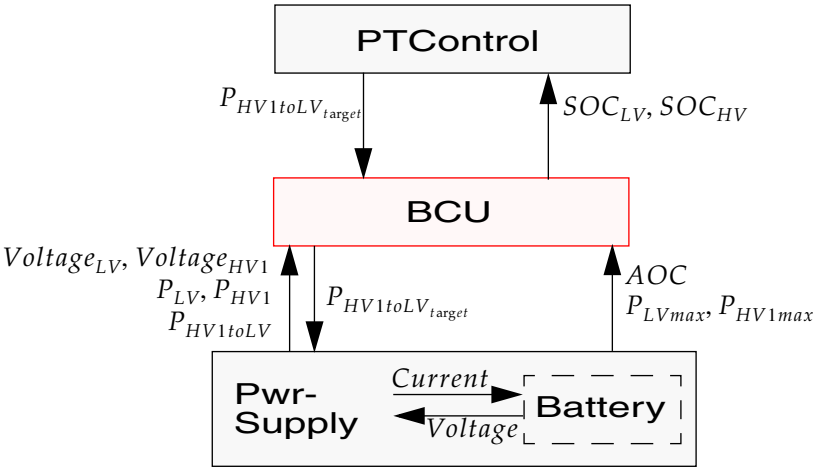


Figure 15.34: BCU model (for two electric circuits)

PowerTrain.BCU.Kind = *KindStr VersionId*

Selection of the battery control unit subsystem to use. The powertrain library provides the following submodels:

ModelName	KindStr	Description
Low Voltage	LV	Battery control unit for power supply with only a low voltage electric circuit
Low Voltage + High Voltage 1	LV_HV1	Battery control unit for power supply with a low voltage and one high voltage electric circuit H1
Low Voltage + High Voltage 1 + High Voltage 2	LV_HV1_HV2	Battery control unit for power supply with a low voltage and two high voltage electric circuits H1 and H2

Example `PowerTrain.BCU.Kind = LV 1`

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tPTBatteryCU_CfgIF*:

Input Variable	Unit	Description
BattLV.SOC_min BattHV.SOC_min	%	Low / high voltage battery minimum admitted state of charge
BattLV.SOC_max BattHV.SOC_max	%	Low / high voltage battery maximum admitted state of charge

Input Variable	Unit	Description
BattLV.Capacity BattHV.Capacity	Ah	Low / high voltage battery capacity
BattLV.Voltage BattHV.Voltage	V	Low / high voltage battery open circuit voltage

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tPTBatteryCU_IF*:

Input Variable	Unit	Description
Ignition	bool	Vehicle ignition
Pwr_LV	W	Total electric power (generators and consumers) on low voltage electric circuit
Pwr_HV1 Pwr_HV2	W	Total electric power (generators and consumers) on high voltage 1 / 2 electric circuit
Voltage_LV	V	Low voltage electric circuit instantaneous voltage
Voltage_HV1 Voltage_HV2	V	High voltage electric circuit 1 / 2 instantaneous voltage
Pwr_HV1toLV	W	Instantaneous transferred electric power from high voltage 1 circuit to low voltage circuit
Pwr_HV1toLV_trg	W	Target transferred electric power from high voltage 1 circuit to low voltage circuit
Pwr_HV1toHV2	W	Instantaneous transferred electric power from high voltage 1 circuit to high voltage 2 circuit
Pwr_HV1toLV_max	W	Maximum possible transferred electric power from high voltage 1 circuit to low voltage circuit
Pwr_HV1toHV2_max	W	Maximum possible transferred electric power from high voltage 1 circuit to high voltage 2 circuit
Eta_HV1toLV		DC/DC efficiency from high voltage 1 circuit to low voltage circuit
Eta_HV1toHV2		DC/DC efficiency from high voltage 1 circuit to high voltage 2 circuit
BattLV.Current BattHV.Current	A	Low / high voltage battery electric current
BattLV.AOC BattHV.AOC	Ah	Low / high voltage battery instantaneous amount of charge
BattLV.Temp BattHV.Temp	K	Low / high voltage battery temperature
BattLV.Energy BattHV.Energy	kWh	Low / high voltage battery remaining energy capacity
BattLV.Pwr_max BattHV.Pwr_max	W	Low / high voltage battery maximum charge/ recharge power

Output Variable	Unit	Description
Status	-	BCU status
SOC_LV SOC_HV	%	Low / high voltage battery instantaneous state of charge

Output Variable	Unit	Description
SOH_LV SOH_HV	%	Low / high voltage battery state of health
BattLV.Current BattHV.Current	A	Low / high voltage battery electric current
BattLV.AOC BattHV.AOC	Ah	Low / high voltage battery instantaneous amount of charge
BattLV.Temp BattHV.Temp	K	Low / high voltage battery temperature
BattLV.Energy BattHV.Energy	kWh	Low / high voltage battery remaining energy capacity
BattLV.Pwr_max BattHV.Pwr_max	W	Low / high voltage battery maximum charge/ recharge power
Pwr_HV1toLV_trg	W	Target transferred electric power from high voltage 1 circuit to low voltage circuit

15.4 Drive Sources

Each powertrain architecture consists of a certain number of elements that provide a torque at their output shafts (combustion engine and electric motors) and elements that transform this torque by their ratios, efficiencies, inertias etc. (clutches and gearboxes). The CarMaker vehicle GUI groups all elements that are involved at providing and transforming the input torque of the same differential or wheel into Drive Sources. Each powertrain model consists of at least one and at most four Drive Sources. The number of components in a Drive Source differs between different powertrain architectures.

This chapter describes all component models of the Drive Sources.

15.4.1 Engine

Overview

The engine model is primarily concerned with engine torque. It provides the torque to the clutch input shaft. The engine rotation itself is calculated externally by the following clutch.

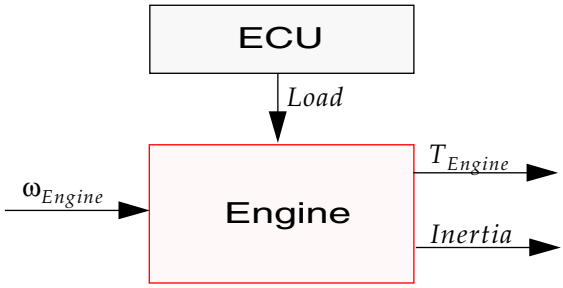


Figure 15.35: Engine model

As shown in Figure 15.35 the main task of the engine model is to act as a torque source. The output T_{Engine} of the engine model can be calculated with different approaches. This is why several subsystem types for the engine model are distinguished.

PowerTrain.Engine.Kind = *KindStr VersionId*

Selection of the engine subsystem to use. The powertrain components library provides the following engine torque models:

ModelName	KindStr	Description
Look-Up Table	Mapping	Look-Up tables are used to determine the engine torque depending on engine load and engine speed and including fuel consumption model
Characteristic Value	Linear	Simple engine torque model
DVA	DVA	Engine torque is modified via DVA access

```
PowerTrain.Engine.Kind = Mapping 1
```

Parameter Engine

The engine submodels that are provided by the powertrain components library have some parameters in common.

PowerTrain.Engine.I = *value*

Inertia of the engine. Default: 0.07 kgm^2.

PowerTrain.Engine.Orientation = *Kind*

Specifies the engine mounting orientation. Possible orientations are:

- *Long*: Engine crankshaft axis is parallel to the longitudinal vehicle axis.
  The engine propulsion torque is supported to the engine body around the x-axis.
- *Trans*: Engine crankshaft axis is perpendicular to the longitudinal vehicle axis (default).
  The engine propulsion torque is supported to the engine body around the y-axis.

Parameter Fuel tank

PowerTrain.Engine.TankVolume = *value*

Specifies the total volume of the fuel tank. Default: 50l.

PowerTrain.Engine.FuelLevel = *value*

Specifies the initial fuel tank level. Default: 80%.

PowerTrain.Engine.FuelDensity = *value*

Density of the power fuel. Unit: kg/l (corresponds to 1000 kg/m^3). Default: 0.75 kg/l.

PowerTrain.Engine.FuelHeatValue = *value*

Heat value of the power fuel. Default: 11.3 kWh/kg.

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tPTEngineCfgIF*:

Input Variable	Unit	Description
FuelDensity	kg/m^3	Fuel density

Output Variable	Unit	Description
rotv_off	rad/s	Engine off rotation speed
rotv_max	rad/s	Engine maximum rotation speed
rotv_idle	rad/s	Engine idle speed
rotv_opt	rad/s	Engine optimum speed with optim. consumption
TrqFull	Nm	1D-Lookup table of maximum engine full torque
TrqDrag	Nm	1D-Lookup table of maximum engine drag torque
TrqOpt	Nm	1D-Lookup table of engine torque with optimum consumption as function of engine speed

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tPTEngineIF*:

Input Variable	Unit	Description
Ignition	bool	Vehicle ignition
FuelCutOff	bool	Flag if fuel is cut-off
Load	-	Load signal for engine
FuelLevel	%	Current fuel level in tank
rot	rad	Current engine output shaft rotation angle
rotv	rad/s	Current engine output shaft rotation speed

Output Variable	Unit	Description
Trq	Nm	Current engine torque at driven shaft
Inert	kgm^2	Engine inertia at driven shaft
FuelFlow	l/s	Current engine fuel flow

Engine model “Look-Up Table”

This engine torque model uses characteristics to describe the behavior of the engine torque depending on engine speed and load.

A measured standard engine torque map usually does not cover parts below minimal and above maximal engine speed. For the usage with CarMaker it is essential that there are values given above and below those boundaries.

Figure 15.36 shows an engine torque map ‘prepared’ for the usage with CarMaker with extended boundaries. The standard speed range of this engine is 500 to 7000 rpm. Above 7000 rpm usually the throttle cutoff applies. This is realized through ramping down the full load characteristics to zero at 8000 rpm and extrapolating the drag load characteristic to 8000 rpm. Interim values for engine load should be equally spaced between full and drag load.

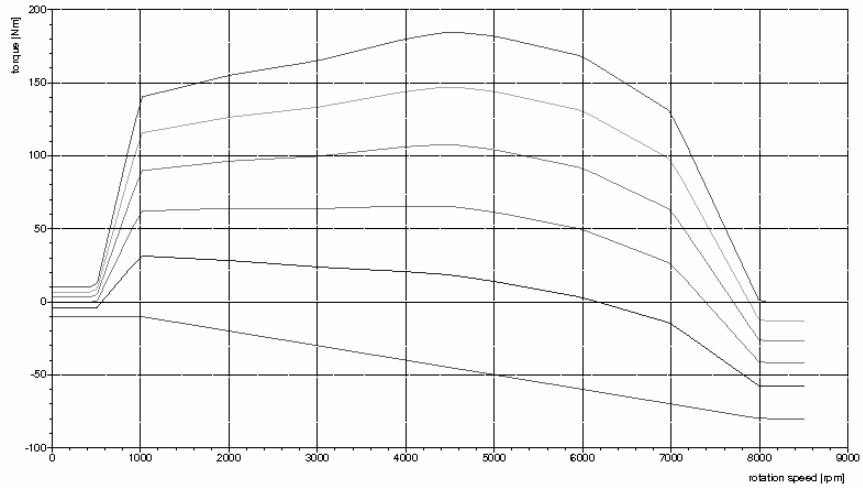


Figure 15.36: Sample engine torque map

Parameter Look-Up Table

PowerTrain.Engine.Mapping.Kind = *KindStr*

This model supports two subsystems to be selected.

ModelName	KindStr	Description
1D Look-Up Table	DragFullLoad	Two characteristic lines: one for full load and one for drag load (depending from engine speed)
2D Look-Up Table	Linear2D	A two dimensional engine characteristic map is used. Depends on engine speed and load.

PowerTrain.Engine.nIdle = *value*

Specifies the engine idle speed. Default: 800rpm.

PowerTrain.Engine.rotvOff = *value*

Specifies the engine off speed. Default: 500rpm.

PowerTrain.Engine.TrqKl15Off = *value*

Engine drag torque with ignition off. Default -80Nm.

PowerTrain.Engine.tBuildUp = *value*

Specifies the build-up time of engine gas. The delayed build-up is modeled by a PT1 filter. Default: 0.

Parameter 1D Look-Up Table

If *1D Look-Up Table* is selected the following parameters have to be specified:

PowerTrain.Engine.Exponent = *value*

Transition from full load to drag load characteristic with a exponential function depending on engine target load.

- load = 0 ... 1.0
- Exponent = 1 → linear
- Exponent > 1→ parabolic
- Exponent < 1 → root shaped

$$Trq = TrqDrag \cdot (1.0 - load^{Exponent}) + TrqFull \cdot load^{Exponent}$$

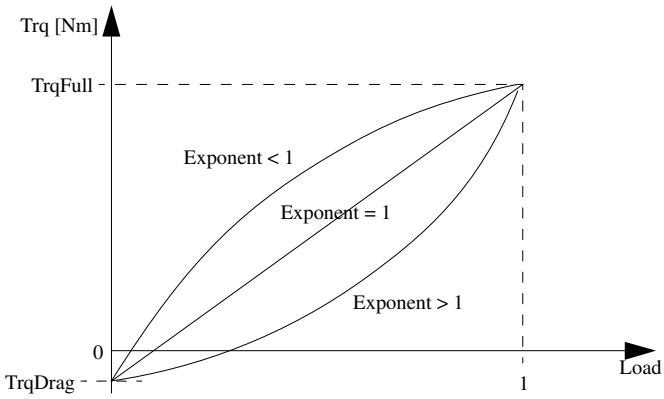


Figure 15.37: load characteristic with an exponential function

PowerTrain.Engine.DragPower.Amplify = *value*
PowerTrain.Engine.FullLoadPower.Amplify =*value*

Amplifies the output of the engine characteristic by a given factor. Default: 1.

PowerTrain.Engine.DragPower = *Table*
PowerTrain.Engine.FullLoadPower =*Table*

Drag power (full load power) characteristic.

Syntax

Infofile table mapping with 2 columns
<engine speed>[rpm] <engine torque> [Nm]

Example

```
PowerTrain.Engine.DragPower:    PowerTrain.Engine.FullLoadPower:

500.0    -10.0                  500.0        10.0
1000.0    -10.0                 1000.0      140.0
...                             ...
6000.0    -60.0                 6000.0      168.0
7000.0    -80.0                 7000.0        0.0
```

Parameter 2D Look-Up Table

If *2D Look-Up Table* is selected the following parameters have to be specified:

PowerTrain.Engine.Mapping.Amplify =*value*

Amplifies the output of the engine characteristic by a given factor. Default: 1.

PowerTrain.Engine.Mapping.Data : *Table*

Two dimensional characteristic for the engine torque mapping. Specifies blocks for equal speed and vary gas from min to max.

Syntax

Infofile table mapping with 3 columns
<engine speed> <engine load> <engine torque>

Example

```
PowerTrain.Engine.Mapping.Data:
500     0.0    -10
500     1.0     10
...
1000    0.0     -10
1000    1.0     120
...
2000    0.0     -20
2000    1.0     145
```

In the GUI the mapping of the kind *2D Look-Up Table* can be displayed in three different ways (see Figure 15.38). With the right mouse click in the figure the mapping is illustrated with lines, points or lines&points.

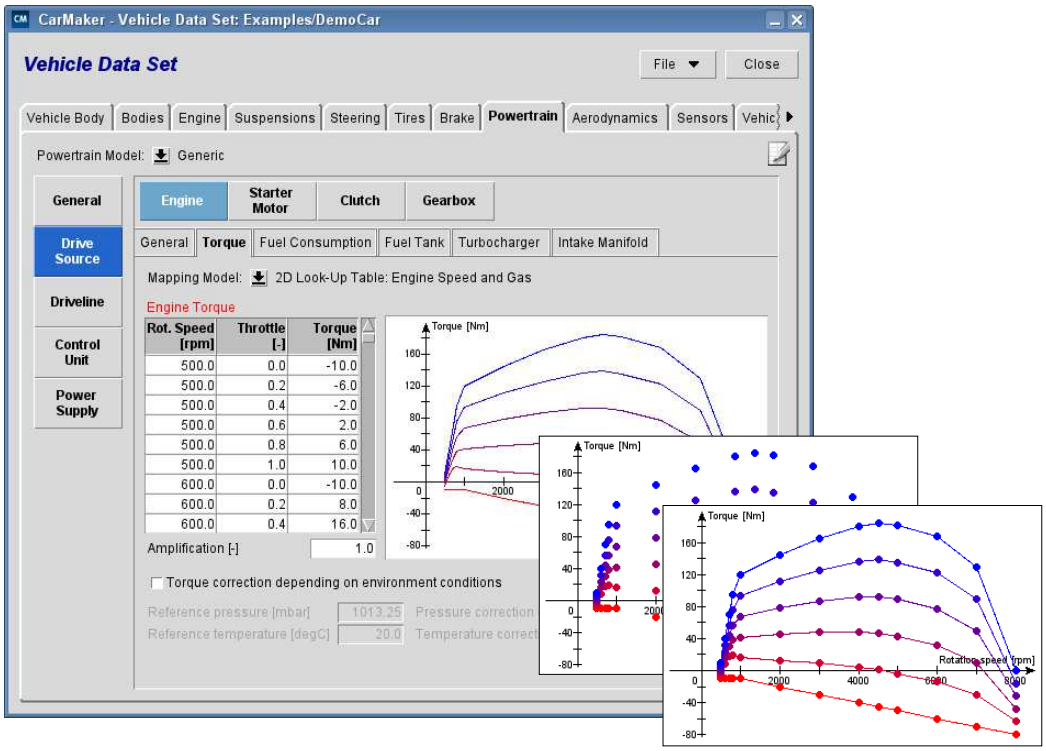


Figure 15.38: Engine mapping of “2D Look-Up Table” in the GUI

Engine Power Correction

For the engine torque model *Mapping*, additionally it is possible to correct the current engine power or torque depending on the current environment conditions (air temperature T_E and air pressure p_E).

$$Trq_{Eng}^{corr} = fac(T_E, p_E) \cdot Trq_{Eng} \quad \text{(EQ 144)}$$

The correction factor is determined as follows:

$$fac(T_E, p_E) = \frac{1}{\left(\frac{p_{ref}}{p_E}\right)^{c_p} \cdot \left(\frac{T_E + 273}{T_{ref} + 273}\right)^{c_T}} \quad \text{(EQ 145)}$$

The parameters p_{ref}, T_{ref}, c_p and c_T describe the conditions, at which the engine torque mapping is valid and with which correction standard it was measured. For example for DIN 70020 the parameters are: $p_{ref} = 1013mbar$, $T_{ref} = 20degC$, $c_p = 1$ and $c_T = 0.5$. The parameters could also differ for diesel or petrol engine, for naturally aspirated or turbo-charged engine.

The engine power/torque correction is not considered by the ECU and PTControl models.

PowerTrain.Engine.PwrCorr.Active = *bool*

Activates the engine power/torque correction depending on environment conditions.
Default: 0.

PowerTrain.Engine.PwrCorr.Pres_ref = *value*

Specifies the absolute reference pressure. Corresponds to p_{ref} in (EQ 145).
Default: 1013.25mbar.

PowerTrain.Engine.PwrCorr.Temp_ref = *value*

Specifies the reference temperature. Corresponds to T_{ref} in (EQ 145). Default: 20degC.

PowerTrain.Engine.PwrCorr.cPres = *value*

Specifies the pressure correction coefficient. Corresponds to c_p in (EQ 145). Default: 1.0.

PowerTrain.Engine.PwrCorr.cTemp = *value*

Specifies the temperature correction coefficient. Corresponds to c_T in (EQ 145).
Default: 0.5.

Fuel consumption mapping

For the engine torque model *Mapping*, additionally a fuel consumption map can be used for the calculation of fuel consumption. The fuel consumption can either be defined by an absolute or a specific fuel consumption mapping.

For the interface to the ECU and PTControl model the volume flow $\dot{v}_B$ with the unit [l/s] is required.

Using a specific fuel consumption mapping, $\dot{v}_B$ is calculated as follows:

$$\dot{v}_F = \frac{\dot{m}_F(\omega_{Eng}, Trq_{Eng}) \cdot |P_{Eng}|}{\varsigma_F \cdot 3,6 \cdot 10^9} \qquad \text{(EQ 146)}$$

with the specific mass flow $\dot{m}_F(\omega_{Eng}, Trq_{Eng})$ from the mapping, the fluid density ς_F and engine output power P_{Eng}. To ensure a positive volume flow in the case of negative engine torque, the absolute value of the engine power is taken.

Using an absolute fuel consumption mapping, $\dot{v}_B$ can be calculated directly as follows:

$$\dot{v}_F = \frac{\dot{m}_F(\omega_{Eng}, Trq_{Eng})}{10^3 \cdot \varsigma_F} \qquad \text{(EQ 147)}$$

with the mass flow $\dot{m}_F(\omega_{Eng}, Trq_{Eng})$ from the mapping. Thus the calculation of fuel consumption is independent on engine power and provide a reliable volume flow with idle engine speed.

PowerTrain.Engine.FuelConsume = *bool*

Activate the fuel consumption mapping and the calculation of fuel consumption. Default: 0.

PowerTrain.Engine.FuelMap.Kind = *Kind*

Consumption mapping kind. Possible mapping kinds are `Specific` (specific fuel consumption mapping) or `Absolute` (absolute fuel consumption mapping). Default: `Specific`.

PowerTrain.Engine.FuelMap.Amplify = *value*

Amplifies the output of the fuel consumption map by a given factor. Default: 1.

PowerTrain.Engine.FuelMap.Data: *Table*

Two dimensional characteristic for the fuel consumption mapping. Specifies blocks for equal speed and vary engine torque from min to max.

Syntax

Infofile table mapping with 3 columns

<engine speed> [rpm]	<engine torque> [Nm]	<specific> [g/kWh] or <absolute> [g/s] fuel consumption

Example

```
PowerTrain.Engine.FuelMap.Data:

   500.0     12.0      600.0
   1000.0    10.0      600.0
   1000.0    30.0      500.0
   1000.0    50.0      400.0
   1000.0    68.0      360.0
   1000.0    95.0      300.0
   1000.0    120.0     270.0
   1500.0    5.0       600.0
   1500.0    20.0      500.0
   1500.0    40.0      400.0
   1500.0    60.0      360.0
   1500.0    80.0      300.0
   1500.0    100.0     270.0
   1500.0    120.0     260.0
   1500.0    140.0     260.0
   2000.0    4.0       600.0
```

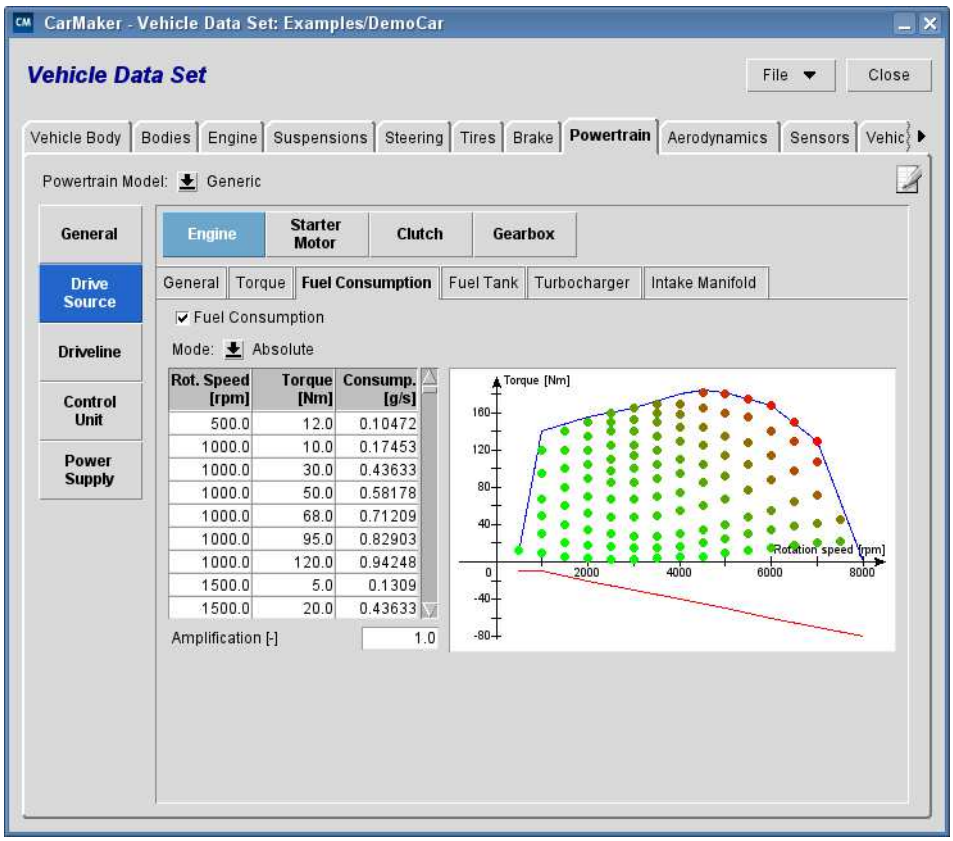


Figure 15.39: Fuel consumption mapping in the GUI

Turbocharger

For the engine torque model *Mapping* the static engine torque from the look-up table can be modeled with a delayed build-up time using a PT1 filter. For an engine with a turbocharger this approach is not exact enough to reproduce the typical turbocharger lag, which is due to the time needed by the exhaust system to get the turbocharger to the required speed. Figure 15.40 illustrates the build-up of the engine torque using the turbo model compared to the both other approaches:

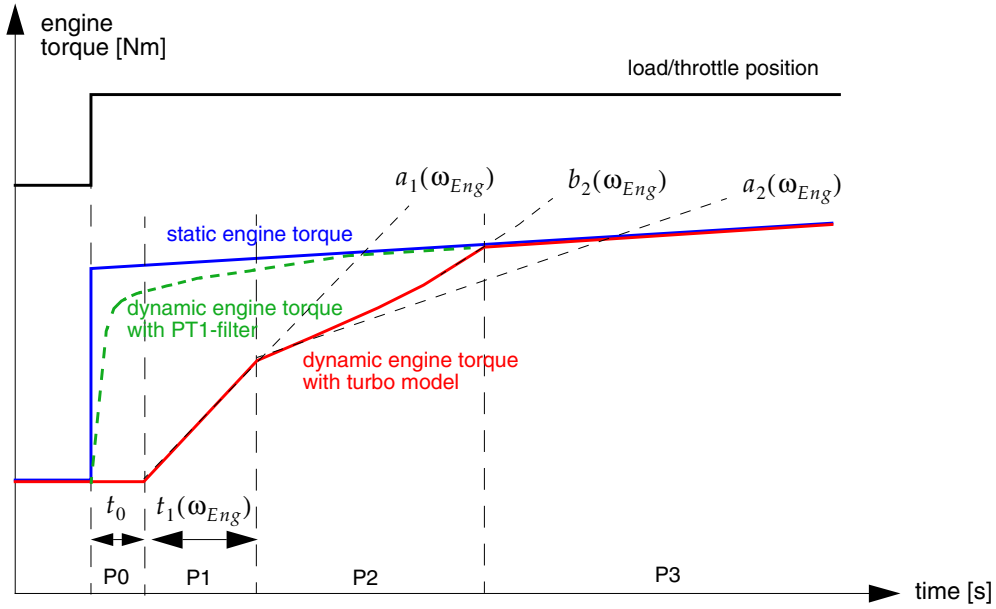


Figure 15.40: Engine torque build-up with the turbo model

The torque build-up in the turbo model is divided into four phases.

- *Phase P0*: This represents the dead time, where the initial torque is held at a constant value $T_{Eng} = T_{init0}$. The dead time $t_0(\omega_{Eng})$ depends on the engine speed.
- *Phase P1*: In this phase the torque is built-up with a linear gradient, depending on engine speed: $T_{Eng} = T_{init1} + \Delta t_1 \cdot a_1(\omega_{Eng})$. The duration time $t_1(\omega_{Eng})$ of this phase depends on the engine speed.
- *Phase P2*: In this phase the torque is build-up with a quadratic function, depending on engine speed: $T_{Eng} = T_{init2} + \Delta t_2 \cdot a_2(\omega_{Eng}) + \Delta t_2^2 \cdot b_2(\omega_{Eng})$. This phase continues until the static engine torque is reached.
- *Phase P3*: In this phase the dynamic engine torque with turbo model and the static engine torque are identical.

The calculation of the dynamic engine torque with the described phases is only used if the load gradient $\Delta load / \Delta t$ exceeds a predefined threshold value to identify a tip-in.

PowerTrain.Engine.Turbo.Active = *bool*

Activate the turbocharger model for the dynamic torque calculation. Default: 0.

PowerTrain.Engine.Turbo.Pedal_slope4TI = *value*

Specifies the threshold value for load gradient to activate the dynamic calculation. Default: 3.0 1/s.

PowerTrain.Engine.Turbo.tPhase_0: *Table*

One dimensional characteristic for the dead time $t_0(\omega_{Eng})$ in phase P0.

Syntax
Infofile table mapping with 2 columns
<engine speed>[rpm] <dead time> [s]

Example

```
PowerTrain.Engine.Turbo.tPhase_0:
1500     0.08
2800     0.05
```

PowerTrain.Engine.Turbo.tPhase_1 : *Table*

One dimensional characteristic for the phase P1 time $t_1(\omega_{Eng})$.

Syntax
Infofile table mapping with 2 columns
<engine speed>[rpm] <phase P1 time> [s]

Example

```
PowerTrain.Engine.Turbo.tPhase_1:
1000     0.4
2500     0.3125
```

PowerTrain.Engine.Turbo.TrqGrad_1 : *Table*
PowerTrain.Engine.Turbo.TrqGrad_2 : *Table*

One dimensional characteristic for the linear gradient in phase P1 $a_1(\omega_{Eng})$ and in phase P2 $a_2(\omega_{Eng})$.

Syntax
Infofile table mapping with 2 columns
<engine speed>[rpm] <gradient> [Nm/s]

Example

```
PowerTrain.Engine.Turbo.TrqGrad_1:
500     100
1100    100
2500    275
```

PowerTrain.Engine.Turbo.Curvature_2 : *Table*

One dimensional characteristic for the curvature in phase P2 $b_2(\omega_{Eng})$.

Syntax
Infofile table mapping with 2 columns
<engine speed>[rpm] <curvature> [Nm/s^2]

Example

```
PowerTrain.Engine.Turbo.Curvature_2:
1000    20
2000    20
```

Intake Manifold Pressure

For the engine torque model *Mapping* a simple model for an intake manifold pressure can be used, for example for a brake booster. This model is considered for a naturally aspirated engine without air precompression. Figure 15.41 illustrates a typical pressure behavior depending on engine speed and throttle position (engine load).

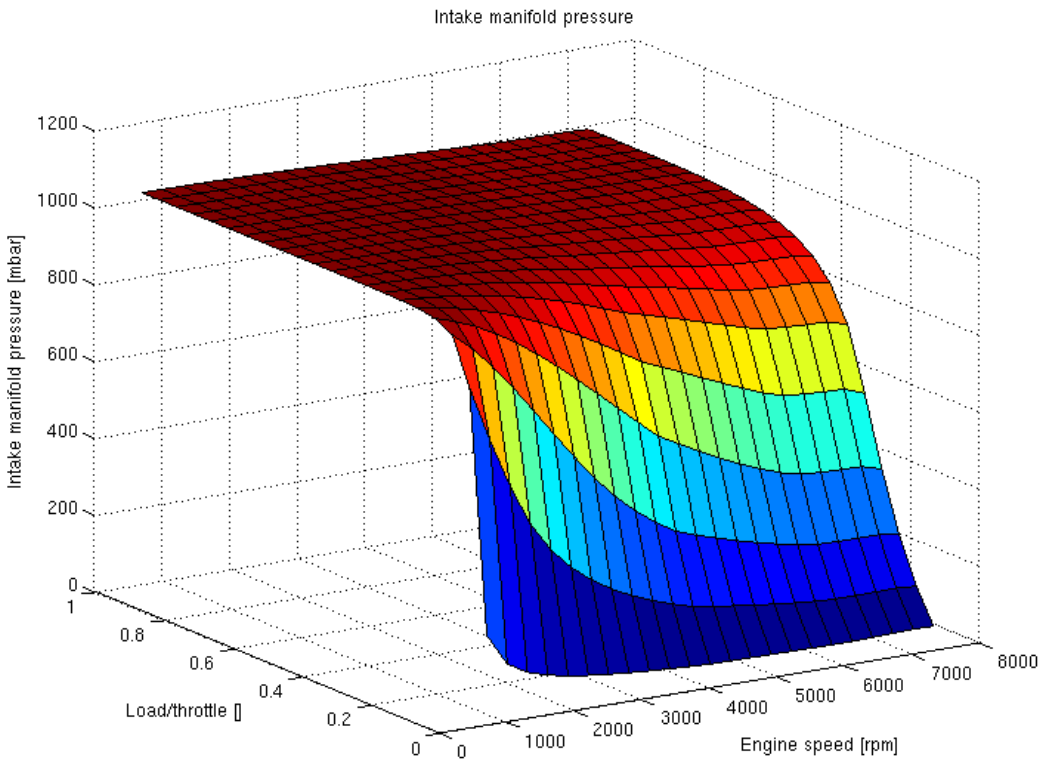


Figure 15.41: Typical intake manifold pressure for naturally aspirated engine

The model supports two kinds of implementation, a dynamic physical model with time integration or via a static 2D-look-up table.

Using the static model, the intake manifold pressure p_{IM} is calculated as follows:

$$p_{IM} = \varsigma_{IM}^{rel} \cdot p_E \cdot \frac{T_{IM}}{T_E} \qquad \text{(EQ 148)}$$

with environment air pressure p_E, environment air temperature T_E and with intake manifold temperature $T_{IM} = T_E + T_{off}$, which is supposed to correspond the environment temperature corrected with an offset.

The relative intake manifold density $\varsigma_{IM}^{rel} = \frac{\varsigma_{IM}}{\varsigma_E} = f(\omega, Load)$ is stored as static 2D-look-up table.

With the dynamic approach, the model uses the following model:

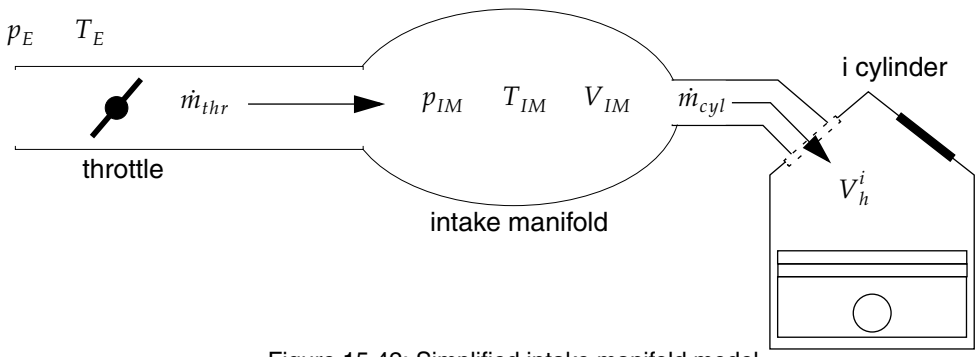


Figure 15.42: Simplified intake manifold model

Using the dynamic model, the intake manifold pressure is determined by integration:

$$p_{IM} = p_{IM}^{0} + \int \dot{p}_{IM} dt \qquad \text{(EQ 149)}$$

The pressure derivation $\dot{p}_{IM}$ is calculated as follows, supposing constant intake manifold temperature:

$$\dot{p}_{IM} = \frac{(\dot{m}_{thr} - \dot{m}_{cyl}) \cdot R \cdot T_{IM}}{V_{IM}} \qquad \text{(EQ 150)}$$

with intake manifold volume V_{IM} and gas constant $R = 287,058 \frac{J}{kgK}$.

The air mass flow after throttle $\dot{m}_{thr}$ is computed with:

$$\dot{m}_{thr} = A_{thr} \cdot \alpha \cdot p_E \cdot \sqrt{\frac{2}{R \cdot T_E}} \cdot \Psi \qquad \text{(EQ 151)}$$

with throttle area A_{thr}, loss coefficient $\alpha = f(Load)$ depending on throttle position and flow coefficient Ψ (equation St. Venant):

$$\Psi\left(\frac{p_{IM}}{p_E}\right) = \sqrt{\frac{\kappa}{\kappa - 1}\left[\left(\frac{p_{IM}}{p_E}\right)^{\frac{2}{\kappa}} - \left(\frac{p_{IM}}{p_E}\right)^{\frac{\kappa+1}{\kappa}}\right]} \qquad 1 \geq \frac{p_{IM}}{p_E} \geq \left(\frac{2}{\kappa+1}\right)^{\frac{\kappa}{\kappa-1}} \qquad \text{(EQ 152)}$$

$$\Psi = \Psi_{crit} = \Psi\left(\left(\frac{2}{\kappa+1}\right)^{\frac{\kappa}{\kappa-1}}\right) \qquad \frac{p_{IM}}{p_E} < \left(\frac{2}{\kappa+1}\right)^{\frac{\kappa}{\kappa-1}}$$

The isentropic exponent of the air is $\kappa = 1,4$.

The air mass flow in the engine cylinders $\dot{m}_{cyl}$ is computed with:

$$\dot{m}_{cyl} = \lambda_N \cdot V_h^i \cdot \frac{\omega}{2\pi} \cdot \frac{p_N}{R \cdot T_N} \qquad \text{(EQ 153)}$$

with air pressure $p_N = 101325 Pa$ and temperature $T_N = 273,15 K$ at standard conditions, engine speed ω, engine volume per revolution V_h^i and relative filling λ_N at standard conditions. The relative filling λ_N is stored as 2D-look-up table, depending on engine speed and the intake manifold pressure (see Figure 15.43).

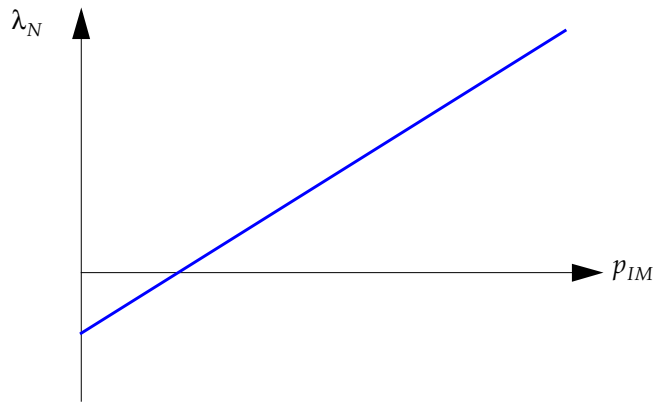


Figure 15.43: Relative filling at constant engine speed

PowerTrain.Engine.IMP.Active = *bool*

Activation of the intake manifold pressure model. Default: 0.

PowerTrain.Engine.IMP.Kind = *string*

Specifies the model calculation kind. Following options are supported:

- *Linear2D*: Using static model with 2D-lookup-table (default).
- *Dynamic*: Using dynamic model with time integration of one DOF model.

PowerTrain.Engine.IMP.Temp_im_off = *value*

Offset temperature in the intake manifold referred to the environment temperature. Corresponds to the parameter T_{off} in the description above. Default: 5K.

Parameter IMP Linear2D

PowerTrain.Engine.IMP.RelRho_im.Amplify = *value*

Amplifies the output of the intake manifold relative density by a given factor. Default: 1.

PowerTrain.Engine.IMP.RelRho_im.Data : *Table*

Two dimensional characteristic for the intake manifold relative density mapping. Corresponds to ς_{IM}^{rel} from (EQ 148).

Syntax

Infofile table mapping with 3 columns

<engine speed>	<load/throttle>	<relative density>
[rpm]	[]	[]

Example

```
PowerTrain.Engine.IMP.RelRho_im.Data:
        700    0      0.225975
        700    0.1    0.946824
        700    0.2    0.977537
        700    0.3    0.981627
        700    0.4    0.982488
        700    0.5    0.982791
        700    0.6    0.982921
        700    0.7    0.982978
        700    0.8    0.98299
        700    0.9    0.982994
        700    1      0.982997
        1000   0      0.145718
        1000   0.1    0.90239
        1000   0.2    0.970052
        1000   0.3    0.979499
        1000   0.4    0.981501
        ...
```

Parameter
IMP Dynamic

PowerTrain.Engine.IMP.Vol_im = *value*

Specifies the intake manifold volume. Corresponds to V_{IM} in (EQ 150). Default: 2.9l.

PowerTrain.Engine.IMP.Vol_eng = *value*

Specifies the total engine cylinder capacity. Used for calculation of V_h^i in (EQ 153). Default: 1.8l.

PowerTrain.Engine.IMP.i_rev = *value*

Specifies the ratio of total engine cylinder capacity used per one revolution. Used for calculation of V_h^i in (EQ 153). Default: 0.5.

PowerTrain.Engine.IMP.Area_thr = *value*

Specifies the area of throttle without loss. Corresponds to A_{thr} in (EQ 151). Default: 14cm^2.

PowerTrain.Engine.IMP.Alpha: *Table*

One dimensional characteristic for the throttle loss coefficient α in (EQ 151). The first column for the load/throttle must begin with 0 and end with 1.

Syntax

Infofile table mapping with 2 columns
<load / throttle>[] <loss coefficient> []

Example

```
PowerTrain.Engine.IMP.Alpha:
   0      0.004
   0.05   0.032
   0.1    0.07
   0.15   0.12
   0.2    0.18
   0.3    0.34
   0.4    0.5
   0.5    0.65
   0.6    0.775
   0.65   0.83
   0.7    0.858
   0.75   0.875
   0.8    0.88
   0.9    0.886
   1.0    0.893
```

PowerTrain.Engine.IMP.Lambda.Amplify = *value*

Amplifies the output of the intake manifold relative filling by a given factor. Default: 1.

PowerTrain.Engine.IMP.Lambda.Data : *Table*

Two dimensional characteristic for the intake manifold relative filling at standard conditions. Corresponds to λ_N from (EQ 153).

Syntax

Infofile table mapping with 3 columns

<engine speed> [rpm]	<intake manifold pressure> [mbar]	<relative filling> []

Example

```
PowerTrain.Engine.IMP.Lambda.Data:
      700       0      -0.04
      700       1013    0.7
      1000      0      -0.04
      1000      1013    0.75
      2000      0      -0.04
      2000      1013    0.8
      3000      0      -0.04
      3000      1013    0.85
      4000      0      -0.04
      4000      1013    0.9
      4500      0      -0.04
      4500      1013    0.88
      5000      0      -0.04
      5000      1013    0.87
      6000      0      -0.04
      6000      1013    0.85
      7000      0      -0.04
      7000      1013    0.75
```

Engine model “Characteristic Value”

This is a simple engine torque model. The engine torque only depends on the load provided from the ECU model. There is no dependency on the rotation speed of the engine (as for a usual engine).

PowerTrain.Engine.nIdle = *value*

Specifies engine idle speed. Default: 800rpm.

PowerTrain.Engine.rotvOff = *value*

Specifies the engine off speed. Default: 500rpm.

PowerTrain.Engine.TrqKl15Off = *value*

Engine drag torque with ignition off. Default -80Nm.

PowerTrain.Engine.SpeedRange = *rotv_min rotv_max*

Specifies the minimal and maximal engine speed. Below and above those speeds the engine torque is regulated down. A ramp function is used. Default: 500 7000 rpm.

Example `PowerTrain.Engine.SpeedRange = 500 7000`

PowerTrain.Engine.PowerRatio =*value*

Specifies the maximum engine torque with full engine load. It is independent of the engine speed. The engine speed is calculated by $\mathrm{Trq} = \mathrm{PowerRatio} \cdot \mathrm{load}$. Default: 300 Nm.

Example `PowerTrain.Engine.PowerRatio = 300`

Engine model “DVA”

This model is *not* a conventional engine model. The user can specify the engine torque by modifying the DVA variable *PT.Engine.DVA.Trq*.

PowerTrain.Engine.TrqKl15Off = *value*

Engine drag torque with ignition off. Default -80Nm.

PowerTrain.Engine.rotv_idle = *value*
PowerTrain.Engine.rotv_off = *value*
PowerTrain.Engine.rotv_max = *value*
PowerTrain.Engine.rotv_opt = *value*
PowerTrain.Engine.TrqFull: *Table*
PowerTrain.Engine.TrqDrag: *Table*
PowerTrain.Engine.TrqOpt: *Table*

For description details of these parameters see below in the description of the next model "User C-code / Simulink Plug-in".

Engine model "User C-code / Simulink Plug-in /FMU"

Following parameters are required for the initialization of the output variables in the interface struct *tPTEngineCfgIF*:

PowerTrain.Engine.rotv_idle = *value*

Specifies the engine idle speed. Default: 800rpm.

PowerTrain.Engine.rotv_off = *value*

Specifies the engine off speed. Default: 500rpm.

PowerTrain.Engine.rotv_max = *value*

Specifies the engine maximum speed. Default: 8000rpm.

PowerTrain.Engine.rotv_opt = *value*

Specifies the engine speed with optimal fuel consumption. Default: 2300rpm.

PowerTrain.Engine.TrqFull: *Table*

Optional. One dimensional characteristic mapping for the maximum engine full torque.

Syntax
Infofile table mapping with 2 columns
<engine speed> [rpm] <engine full torque> [Nm]

Example

```
PowerTrain.Engine.TrqFull:
500.0        10.0
1000.0       140.0
...
```

PowerTrain.Engine.TrqDrag: *Table*

Optional. One dimensional characteristic mapping for the maximum engine drag torque.

Syntax
Infofile table mapping with 2 columns
<engine speed> [rpm] <engine drag torque> [Nm]

Example

```
PowerTrain.Engine.TrqDrag:
500.0        -10.0
1000.0      -10.0
...
```

PowerTrain.Engine.TrqOpt: *Table*

Optional. One dimensional characteristic mapping for engine torque with optimal consumption (minimal specific fuel consumption as function of engine speed).

Syntax
Infofile table mapping with 2 columns
<engine speed> [rpm] <engine optimal torque> [Nm]

Example

```
PowerTrain.Engine.TrqOpt:
500.0      80.0
1000.0    110.0
...
```

15.4.2 Starter motor

Overview

The starter motor is primarily concerned with bringing the engine to its idle speed. Besides it may also act as an additional motor or generator to shift the engines operating point or to recuperate braking energy. The usage of the starter motor strongly depends on the powertrain architecture and the control strategy. The starter model provides the torque to the engine's output shaft based on the load signal provided by the MCU. The starter rotation itself is calculated externally by the powertrain module.

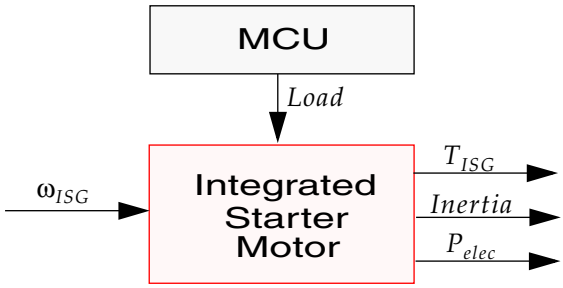


Figure 15.44: Integrated starter motor model

As shown in Figure 15.44 the main task of the starter motor model is to act as a torque source. The output T_{ISG} of the starter motor model can be calculated with different approaches. This is why several subsystem types for the starter motor model are distinguished.

PowerTrain.MotorISG.Kind = *KindStr* *VersionId*

Selection of the starter motor subsystem to use. The powertrain components library provides the following submodels:

ModelName	KindStr	Description
Starter	Starter	Basic starter model that only acts as a motor
Mapping	Mapping	Full electric motor model (see section 'Electric motor model "Mapping"')

Example `PowerTrain.MotorISG.Kind = Mapping 1`

Parameter Starter motor

The starter motor submodels that are provided by the powertrain components library have some parameters in common.

PowerTrain.MotorISG.VoltageLevel = *LevelStr*

Specifies the voltage level (*LV*, *HV1* or *HV2*) of the power supply model to which the starter motor is connected. Default: *LV*.

PowerTrain.MotorISG.I = *value*

Inertia of the starter motor. Default: 1e-6 kgm^2.

PowerTrain.MotorISG.Ratio = *value*

Ratio between starter motor shaft and driven shaft. Default: 4.0.

Interface for the initialization and evaluation function

As the integrated starter generator is an electric motor, its model interface is the same as the interface of an electric motor model (see section 'Electric motor model "Mapping"').

ISG model "Starter"

The ISG model *Starter* is a simple model without AC/DC conversion losses and with identical motor and generator behavior. The maximum motor torque (depending on rotation speed) $T_{Max}(\omega)$ is calculated based on mechanical power P, maximum motor torque T_{Max} and maximum rotation speed ω_{Max}. This characteristic curve consists of three parts:

- For $\omega \leq \omega_{Reference} = \frac{P}{T_{Max}}$

$$T_{Max}(\omega) = T_{Max} \quad \text{(EQ 154)}$$

- For $\omega_{Reference} < \omega \leq \omega_{Max}$

$$T_{Max}(\omega) = \frac{P}{\omega} \quad \text{(EQ 155)}$$

- For $\omega > \omega_{Max}$

$$T_{Max}(\omega) = 0Nm \quad \text{(EQ 156)}$$

The integrated starter motor torque at the current rotation speed is then calculated the following way:

$$T_{ISG} = T_{Max}(\omega) \cdot Load_{ISG} \quad \text{(EQ 157)}$$

Parameter Starter

PowerTrain.MotorISG.Pwr_max = *value*

Maximum power of integrated starter generator. Default: 2000W.

PowerTrain.MotorISG.Trq_max = *value*

Maximum torque of integrated starter generator. Default: 50Nm.

PowerTrain.MotorISG.rotv_max = *value*

Maximum rotation speed of integrated starter generator. Default: 7000rpm.

ISG model "Mapping"

The ISG torque model *Mapping* distinguishes two characteristic maximum torque curves depending on if the starter is used as a motor or a generator. This model is equal to the electric motor torque model *Mapping* described in section 'Electric motor model "Mapping"'.

15.4.3 Electric motor

Overview

The electric motor model is primarily concerned with electric motor torques. It provides motor and generator torques at the driven shaft based on the load signal provided by the MCU model. The motor rotation itself is calculated externally by the powertrain module. As shown in Figure 15.45 the main task of the electric motor model is to act as a torque source.

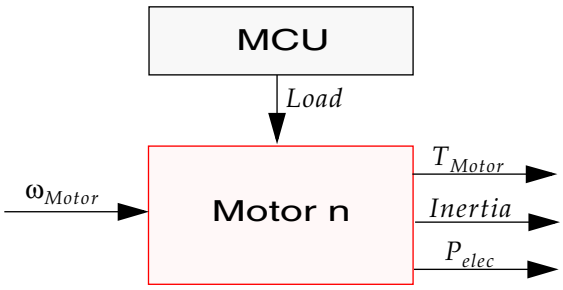


Figure 15.45: Motor model

A powertrain model may contain up to eight electric motors (plus an integrated starter generator). The integrated starter generator is called "MotorISG", the first electric motor is called "Motor", the second "Motor1", the third "Motor2" and so on. These names are used in all Infofile parameters, UAQ and interface signals of the electric motor model, the MCU and PTControl. This chapter uses the parameter names of the first electric motor.

PowerTrain.Motor.Kind = *KindStr* *VersionId*

Selection of the electric motor subsystem to use. The powertrain components library provides the following submodels:

ModelName	KindStr	Description
Mapping	Mapping	Full electric motor model

Example `PowerTrain.Motor.Kind = Mapping 1`

Parameter Electric motor

The electric motor submodels that are provided by the powertrain components library have some parameters in common.

PowerTrain.Motor.VoltageLevel = *LevelStr*

Specifies the voltage level (*LV*, *HV1* or *HV2*) of the power supply model to which the electric motor is connected. Default: *LV*.

PowerTrain.Motor.I = *value*

Inertia of the electric motor. Default: 0.01 kgm^2.

PowerTrain.Motor.Ratio = *value*

Ratio between electric motor shaft and driven shaft. Default: 1.0.

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tPTMotorCfgIF*:

Output Variable	Unit	Description
Level		Electric motor power supply level (LV, HV1, HV2)
Ratio		Ratio between motor shaft and driven shaft
rotv_Mot_max	rad/s	Electric motor maximum motor rotation speed
rotv_Gen_max	rad/s	Electric motor maximum generator rotation speed
TrqMot_max	Nm	1D-Lookup table of electric motor's maximum motor torque
TrqGen_max	Nm	1D-Lookup table of electric motor's maximum generator torque

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tPTMotorIF*:

Input Variable	Unit	Description
Load	-	Load signal for electric motor
rot	rad	Current electric motor output shaft rotation angle
rotv	rad/s	Current electric motor output shaft rotation speed
Voltage	V	Electric voltage at motor

Output Variable	Unit	Description
Trq	Nm	Electric motor torque on driven shaft
Inert	kgm^2	Electric motor inertia on driven shaft
PwrElec	W	Current electric power (generator, motor) of motor

Electric motor model "Mapping"

The electric torque model *Mapping* distinguishes two characteristic maximum torque curves depending on if it is used as a motor or as a generator. Figure 15.46 depicts these characteristic curves.

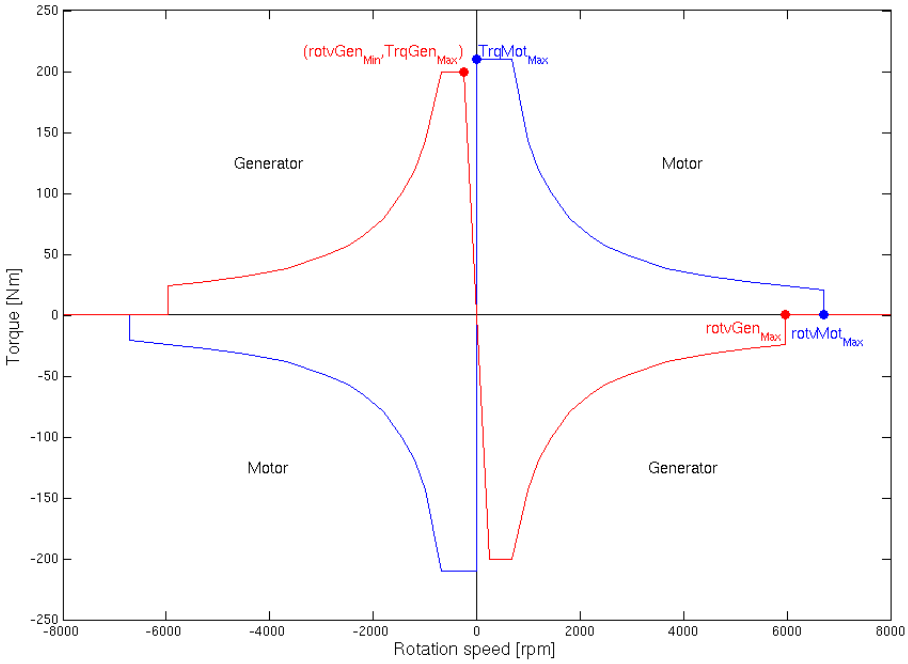


Figure 15.46: Electric motor model "Mapping"

The torque T_{EM} at the output shaft is determined as described by (EQ 158) in which $T_{Max}(\omega)$ represents the maximum motor or generator torque at the current rotation speed depending on the current motor mode.

$$T_{EM} = T_{Max}(\omega) \cdot Load_{EM} \quad \textbf{(EQ 158)}$$

There are two motor torques models available inside motor model *Mapping*: *1D Look-Up Table* and *Characteristic Value*.

The electric power of the motor $P_{ElecMot}$ respectively of the generator $P_{ElecGen}$ are calculated considering the motor efficiency η_{Mot} respectively the generator efficiency η_{Gen}.

$$P_{ElecMot} = \frac{T_{EM} \cdot \omega}{\eta_{Mot}} \quad \textbf{(EQ 159)}$$

$$P_{ElecGen} = T_{EM} \cdot \omega \cdot \eta_{Gen} \quad \textbf{(EQ 160)}$$

Parameter Mapping

The electric motor model *Mapping* has the following parameters, independent of the selected motor torque model.

PowerTrain.Motor.tBuildUp = *value*

Specifies the build-up time of motor load. The delayed build-up is modeled by a PT1 filter. Default: 0.01s.

Parameter Torque

PowerTrain.Motor.Mot.TrqKind = *KindStr*
PowerTrain.Motor.Gen.TrqKind = *KindStr*

This model supports two motor/generator torque definitions to be selected.

ModelName	KindStr	Description
1D Look-Up Table	Map	Characteristic curves for maximum motor/generator torque
Characteristic Value	Value	Defines maximum motor/generator torque via characteristic values

Example

```
PowerTrain.Motor.Mot.TrqKind = Map
```

The electric motor torque model 1*D Look-Up Table* uses characteristic curves to describe $T_{Max}(\omega)$ depending on the rotation speed.

PowerTrain.Motor.Mot.TrqMap: *Table*
PowerTrain.Motor.Gen.TrqMap: *Table*

One dimensional characteristic for the motor/generator torque mapping.

Syntax

Infofile table mapping with 2 columns
<motor/generator speed> <motor/generator torque>

Example

```
PowerTrain.Motor.Mot.TrqMap:
 0.0    210
 682    210
1437    100
...
8000   18
```

PowerTrain.Motor.Mot.TrqMap.Amplify = *value*
PowerTrain.Motor.Gen.TrqMap.Amplify = *value*

Amplifies the output of the motor/generator torque characteristic by a given factor. Default: 1.0.

The electric motor torque model *Characteristic Value* determines $T_{Max}(\omega)$ based on four values: mechanical power P, maximum motor torque T_{Max}, maximum rotation speed ω_{Max} and minimum rotation speed at which the maximum motor torque is available ω_{TrqMax}. For motor models, ω_{TrqMax} is always set to zero.

These characteristic values describe a curve that consists of four parts:

- For $\omega < \omega_{TrqMax}$

$$T_{Max}(\omega) = \frac{T_{Max}}{\omega_{TrqMax}} \cdot \omega \quad \text{(EQ 161)}$$

- For $\omega_{TrqMax} \leq \omega \leq \omega_{Reference} = \frac{P}{T_{Max}}$

$$T_{Max}(\omega) = T_{Max} \quad \text{(EQ 162)}$$

- For $\omega_{Reference} < \omega \leq \omega_{Max}$

$$T_{Max}(\omega) = \frac{P}{\omega} \quad \text{(EQ 163)}$$

- For $\omega > \omega_{Max}$

$$T_{Max}(\omega) = 0Nm \quad \text{(EQ 164)}$$

PowerTrain.Motor.Mot.Pwr_max = *value*
PowerTrain.Motor.Gen.Pwr_max = *value*

Maximum electric motor/generator mechanical power (absolute value). Default: 50000W.

PowerTrain.Motor.Mot.Trq_max = *value*
PowerTrain.Motor.Gen.Trq_max = *value*

Maximum electric motor/generator torque. Default: 200Nm.

PowerTrain.Motor.Mot.rotv_max = *value*
PowerTrain.Motor.Gen.rotv_max = *value*

Maximum electric motor/generator rotation speed. Default: 7000rpm.

PowerTrain.Motor.Gen.rotv_Trq_max = *value*

Minimum rotation speed at which the maximum generator torque is available.
Default: 200rpm.

PowerTrain.Motor.Fric = *value*

Friction coefficient. Default: 1e-4Nms/rad for starter motor, 3e-4Nms/rad for electric motor.

Parameter Efficiency

PowerTrain.Motor.Mot.EtaKind = *KindStr*
PowerTrain.Motor.Gen.EtaKind = *KindStr*

This model supports two efficiency definitions to be selected.

ModelName	KindStr	Description
1D Look-Up Table	Map	Motor/generator efficiency depends on motor rotation speed and motor torque
Characteristic Value	Value	Constant motor/generator efficiency

Example

```
PowerTrain.Motor.Mot.EtaKind = Map
```

PowerTrain.Motor.Mot.EtaMap: *Table*
PowerTrain.Motor.Gen.EtaMap: *Table*

Two dimensional characteristic for the motor/generator efficiency mapping. Specifies blocks for equal normalized speed $\frac{\omega}{\omega_{Max}}$ and vary normalized torque $\frac{T}{T_{Max}}$ from 0 to 1.

Syntax

Infofile table mapping with 3 columns
<norm. rotation speed> <norm. torque> <efficiency>

Example

```
PowerTrain.Motor.Mot.EtaMap:
0.0    0.0   0.95
0.0    1.0    0.95
...
1.0    0.0    0.95
1.0    1.0    0.95
```

PowerTrain.Motor.Mot.EtaMap.Amplify = *value*
PowerTrain.Motor.Gen.EtaMap.Amplify = *value*

Amplifies the output of the motor/generator efficiency characteristic by a given factor. Default: 1.

PowerTrain.Motor.Mot.Eta = *value*
PowerTrain.Motor.Gen.Eta = *value*

Electro mechanic efficiency of motor respectively generator for *PowerTrain.Motor.Mot.EtaKind = Value* respectively *PowerTrain.Motor.Gen.EtaKind = Value*. Default: 1.0.

Electric motor model "User C-code / Simulink Plug-in / FMU"

Following parameters are required for the initialization of the output variables in the interface struct *tPTMotorCfgIF*:

PowerTrain.Motor.VoltageLevel = *LevelStr*

Specifies the voltage level (*LV*, *HV1* or *HV2*) of the power supply model to which the electric motor is connected.

PowerTrain.Motor.Ratio = *value*

Ratio between electric motor shaft and driven shaft. Default: 1.0.

PowerTrain.Motor.Mot.TrqMap: *Table*
PowerTrain.Motor.Gen.TrqMap: *Table*

One dimensional characteristic mapping for the maximum motor/generator torque.

Syntax Infofile table mapping with 2 columns
<motor/generator speed> [rpm] <motor/generator torque> [Nm]

Example
```
PowerTrain.Motor.Mot.TrqMap:
 0.0    210
 682    210
1437    100
...
8000   18
```

Example
```
PowerTrain.Motor.Gen.TrqMap:
 0.0    0
 500    210
1437    100
...
8000   18
```

PowerTrain.Motor.Mot.rotv_max = *value*
PowerTrain.Motor.Gen.rotv_max = *value*

Maximum electric motor/generator rotation speed. Default: 8000rpm.

15.4.4 Clutch

Overview

The clutch calculates the torque transfer from its input shaft to its output shaft considering the clutch pedal or TCU signals.

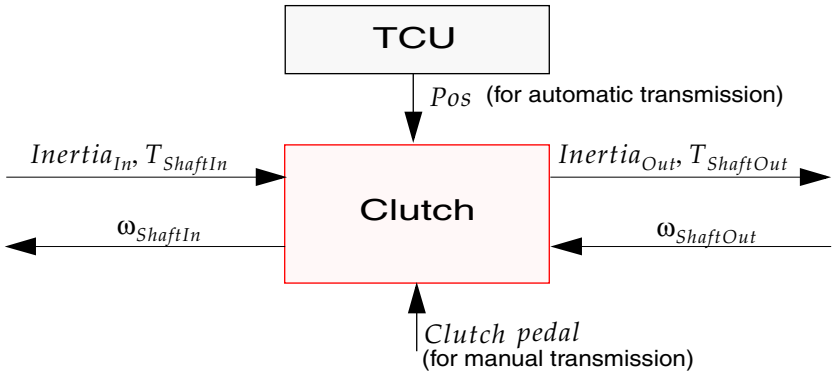


Figure 15.47: Clutch model

In CarMaker, clutch models can be found in two different forms:

- Independent clutches are enclosed models that can be open and closed independently of a gearbox. They are used e.g. as disengaged clutch between combustion engine and electric motor in Parallel P2 or conventional Generic powertrain.
- Integrated clutches are part of a gearbox model. They usually are open only during gear change.

This chapter uses the parameter and signal names of the independent clutch model. The parameter of an integrated clutch within a gearbox is described in the corresponding gearbox model.

PowerTrain.Clutch.Kind = *KindStr* *VersionId*

Selection of the clutch subsystem to use. The powertrain components library provides the following submodels:

ModelName	KindStr	Description
Converter	Converter	Torque converter model (usually used in combination with automatic transmissions)
Friction	Manual	Manual clutch model (usually hand-operated by the driver or used as disengaged clutch to separate the engine from the rest of the powertrain)
Dual-Mass Flywheel	DMF	Manual clutch model with dual-mass flywheel
DVA	DVA	Clutch torque is modified via DVA access
Closed	Closed	No clutch model (input shaft and output shaft firmly aligned)
Open	Open	No clutch model (input shaft and output shaft disconnected)

Example `PowerTrain.Clutch.Kind = Manual 1`

Parameter Clutch

The clutch that are provided by the powertrain components library have some parameters in common.

PowerTrain.Clutch.I_in = *value*

Inertia of the clutch input shaft. Default: 0.0001 kgm^2 (0 for the models *Closed* and *Open*).

PowerTrain.Clutch.I_out = *value*

Inertia of the clutch output shaft. Default: 0.0001 kgm^2 (0 for the models *Closed* and *Open*).

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tPTClutch_CfgIF*:

Output Variable	Description
ClKind	Clutch kind (friction, converter)

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tPTClutchIF* for independent clutch models:

Input Variable	Unit	Description
Pos	-	Clutch target position
rot_out	rad	Current clutch output shaft rotation angle
rotv_out	rad/s	Current clutch output shaft rotation speed
Trq_in	Nm	Current clutch input shaft torque
Inert_in	kgm^2	Clutch input shaft inertia

Output Variable	Unit	Description
rot_in	rad	Current clutch input shaft rotation angle
rotv_in	rad/s	Current clutch input shaft rotation speed
Trq_out	Nm	Current clutch output shaft torque
Inert_out	kgm^2	Clutch output shaft inertia
i_TrqIn2Out	-	Current ratio clutch input shaft torque to output shaft torque (estimated)
Trq_SuppInert	Nm	Support torque of inertia

Clutch model "Friction"

A manual clutch in reality consists of two plates a friction plate and a contact pressure plate. The transmissible torque depends of the contact pressure and the speed difference between the two plates. In a real time environment with fixed step integrators modeling a technical system like a clutch is a challenging task. With conventional methods usually problems occur when transiting from slip to the singularity stick and vice versa. For a better computation stability the modeling approach of the clutch in this powertrain is sightly more complex.

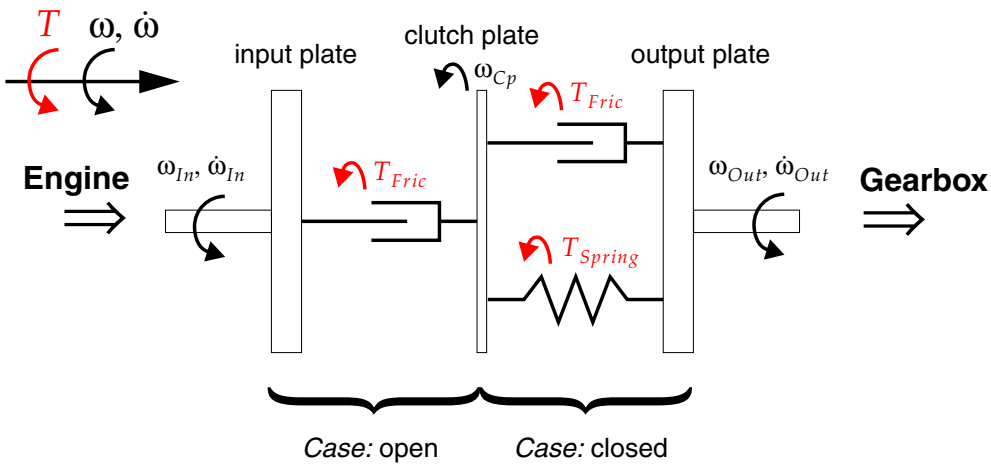


Figure 15.48: Structure of Clutch Model “Friction”

Figure 15.48 shows the non physical approach of this model which uses three plates for transmitting the torque from the engine to the gearbox. The plates from engine side to gearbox side are called “input-”, “clutch-” and “output-plate”. The “input-” and the “clutch-plate” form a system with pure friction which is used to cover the case clutch is opened respectively is slipping. The remaining two plates with pure friction and a spring element form the counterpart for the closed clutch.

- Calculation of the friction torque:

$$\Delta\dot{\omega}_{In2Out} = \dot{\omega}_{In} - \dot{\omega}_{Out} \quad \text{(EQ 165)}$$

$$T_{Fric} = f(\Delta\dot{\omega}_{In2Out}) \quad \text{(EQ 166)}$$

Figure 15.49 shows the function of the friction torque T_{Fric} dependent on the slope (friction coefficient) and the maximal torque value.

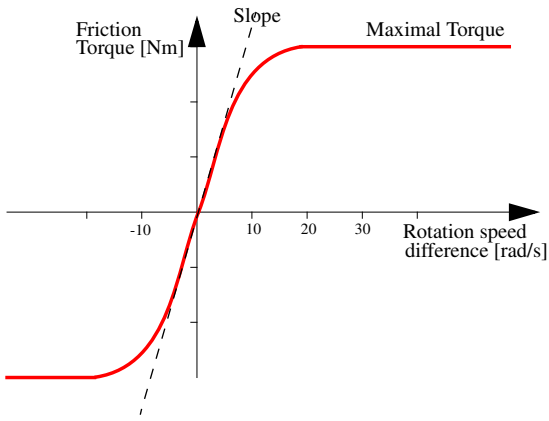


Figure 15.49: Friction Torque

- Calculation of the spring torque:

$$\Delta\omega_{Cp2Out} = \omega_{Cp} - \omega_{Out} \quad \text{(EQ 167)}$$

$$T_{Spring} = c \cdot \Delta\omega_{Cp2Out} \quad \text{(EQ 168)}$$

To differentiate between the case “clutch open“ and the case “clutch closed“, the normalized clutch pedal position λ is needed:

$$\lambda = \frac{PedalPos - ConnectPos}{DisconnectPos - ConnectPos} \quad \text{(EQ 169)}$$

Below `ConnectPos` the clutch is fully closed and above `DisconnectPos` no torque is transmitted.

- Calculation of the clutch torque for the case “clutch open“:

  The clutch is open if the normalized clutch pedal position $\lambda > 0$. For this case the total clutch torque is:

$$T_{In2Out} = T_{Fric} \quad \text{(EQ 170)}$$

- Calculation of the clutch torque for the case “clutch closed“:

  The clutch is closed if the normalized clutch pedal position $\lambda = 0$. For this case the total clutch torque is:

$$T_{In2Out} = T_{Fric} + T_{Spring} \quad \text{(EQ 171)}$$

  If T_{In2Out} exceeds the maximal torque, the clutch begins to slip.

- Limitation of the clutch torque:

  The resulting clutch moment depends on the normalized clutch pedal position and the maximal torque:

$$T_{In2Out} = MIN(T_{In2Out}, T_{Max}) \quad \text{(EQ 172)}$$

$$T_{In2Out} = T_{In2Out} \cdot (1 - \lambda) \quad \text{(EQ 173)}$$

Parameter Friction

PowerTrain.Clutch.ConnectPos = *value*

At this clutch position the clutch starts slipping when opening. Default: 0.3 [0.. 1].

Example `PowerTrain.Clutch.ConnectPos = 0.3`

PowerTrain.Clutch.DisconnectPos = *value*

At this clutch position the clutch starts transferring torque when closing.
Default: 0.7 [0 .. 1].

Example `PowerTrain.Clutch.DisconnectPos = 0.7`

PowerTrain.Clutch.Trq_max = *value*

Sets maximum transmissible torque equally for both cases, clutch is slipping and clutch is closed. Default: 300 Nm.

Example `PowerTrain.Clutch.Trq_max = 300`

PowerTrain.Clutch.slope = *value*

Sets the slope in the origin for the function of the friction torque. The slope is equivalent to the friction coefficient. Default: 30 Nms/rad.

Example `PowerTrain.Clutch.slope = 30`

PowerTrain.Clutch.c.Kind =*KindStr*

Specifies the kind of clutch spring stiffness. Possible values are:

- Linear: linear spring stiffness coefficient (default)
- Curve: non-linear 1D look-up table for clutch spring torque

PowerTrain.Clutch.c = *value*

Sets the spring constant. Default: 1500 Nm/rad.

Example `PowerTrain.Clutch.c = 1500`

PowerTrain.Clutch.c.Data: *Table [Angle Torque]*

One dimensional characteristic for the clutch spring torque mapping.

Syntax Infofile table mapping with 2 columns
<deflection angle [deg]> <spring torque [Nm]>

Example
```
PowerTrain.Clutch.c.Data:
 -20    -120
   0    0
  14    25
  25    200
```

PowerTrain.Clutch.c.Amplify = *value*

Specifies the amplification factor of spring torque map. Default: 1.0.

Clutch model "Dual-Mass Flywheel"

The Dual-Mass Flywheel (DMF) consists of a primary side, which is connected to the engine output shaft and a secondary side, which is connected to the clutch plate. The torque is transmitted from the primary side to the secondary side via a relatively soft spring and friction shims.

The task of the DMF is to damp the vibrations of a combustion engine.

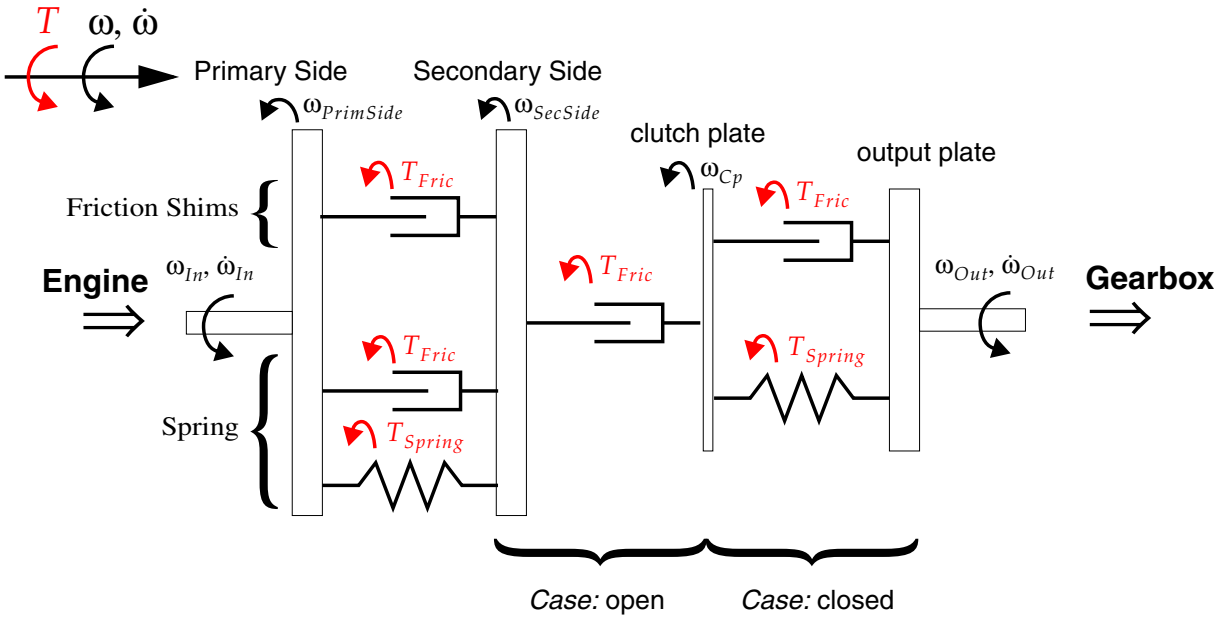


Figure 15.50: Structure of the “Dual-Mass Flywheel” Clutch model

Due to the friction shims there is a constant friction torque between the primary and secondary side.

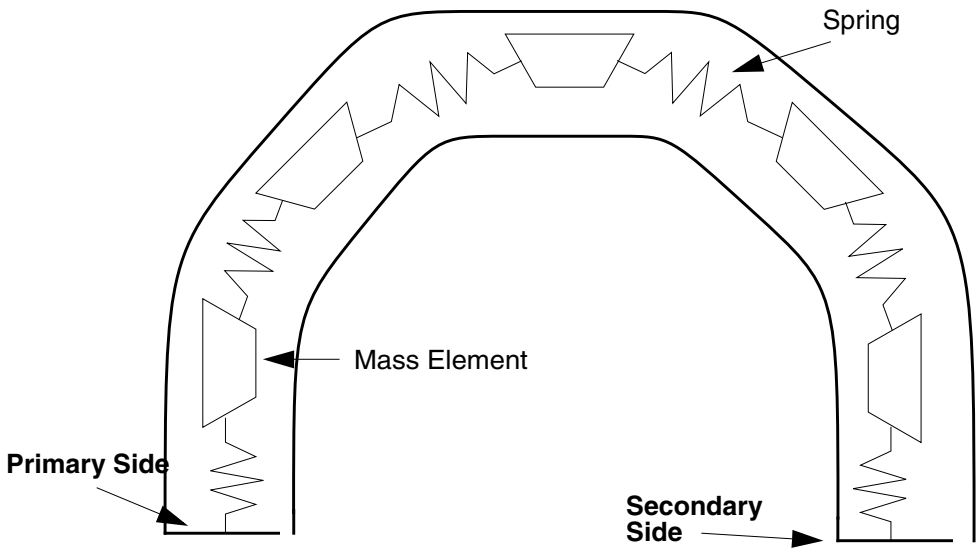


Figure 15.51: Abstraction of the DMF-Spring

The spring can be viewed as several mass elements, connected with massless springs. This way the interaction between different sections of the spring can be calculated.

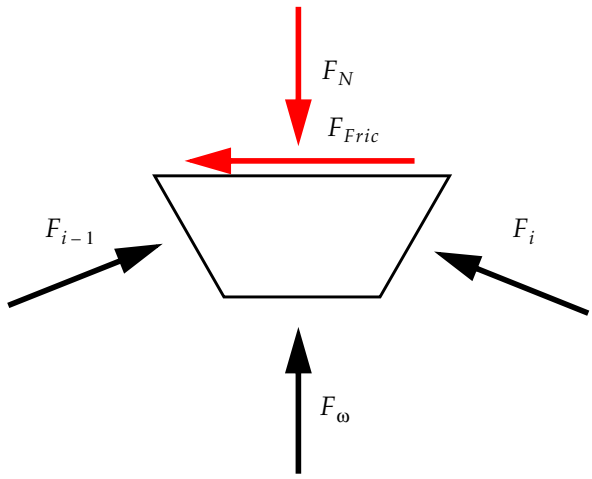


Figure 15.52: Force equilibrium on a mass element

The forces acting on one mass element are calculated with the following equations.
Due to the rotational speed of the engine and the DMF there is a centrifugal force on the spring element:

$$F_\omega = m_{Spring} \cdot \omega^2 \cdot r_m \qquad \textbf{(EQ 174)}$$

with:

m_{Spring} : mass of the spring element
$\omega^2 = 2 \cdot \pi \cdot n_{Engine}$: rotational speed of the DMF
r_m : middle radius of the spring

The spring forces acting on the mass element have both rates in the same direction as the centrifugal force. This can be calculated with:

$$F_{i,N} = F_i \cdot \sin\alpha \qquad \textbf{(EQ 175)}$$

Angle α varies only with the number of elements and is assumed not to change due to the compression of the spring elements:

The normal force acting on the spring can be determined:

$$F_N = F_{i,N} + F_{i+1,N} + F_\omega \qquad \textbf{(EQ 176)}$$

So the friction torque on a mass element is:

$$T_{fric} = \mu \cdot r_a \cdot F_N \qquad \textbf{(EQ 177)}$$

with:

μ : friction coefficient between mass element and spring canal
r_a : outer radius of the spring

To handle the stick-slip behavior near zero the friction torque is multiplied with:

$$T_{fric} = T_{fric} \cdot \tanh\left(\frac{2|\Delta\omega|}{\omega_0}\right) \quad \textbf{(EQ 178)}$$

with:
$\Delta\omega$: relative rotational velocity between spring element and primary side
ω_0 : reference rotational velocity for build-up of the full friction torque

The equation of motion for each spring element results to:

$$\dot{\omega} \cdot \frac{I_{Spring}}{n_{mass}} = F_{i,T} - F_{i+1,T} - T_{fric} \cdot \mathrm{sgn}(\Delta\omega) \quad \textbf{(EQ 179)}$$

$$\text{with } F_{i,T} = F_i \cdot \cos\alpha \quad \textbf{(EQ 180)}$$

Due to the fact that some spring elements can be sliding while others can't, the stiffness of the spring can vary with the rotational speed of the engine. Also the hysteresis increases with increasing engine speed.

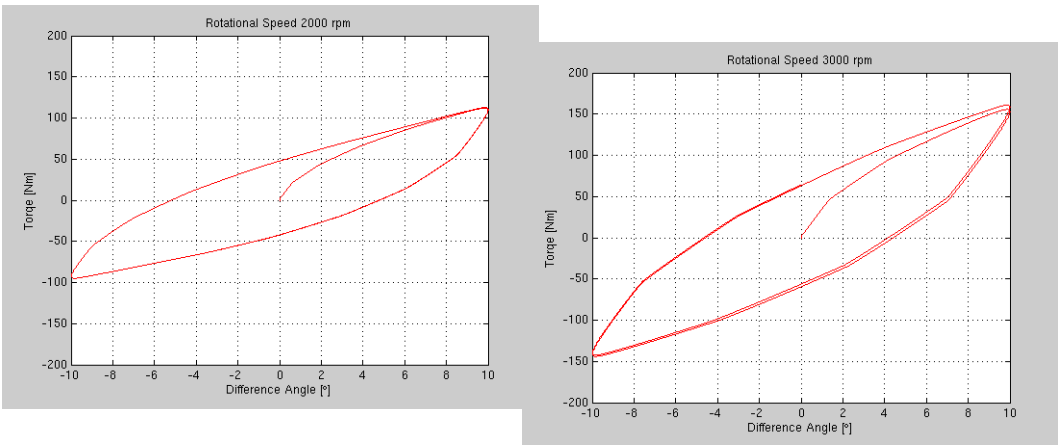


Figure 15.53: Comparison of spring curve at 2000 and 3000 rpm

Parameter Dual-Mass Flywheel

Following parameters describe only the DMF part of the clutch. The right clutch friction part is described in the clutch model "Friction", see section 'Clutch model "Friction"'.

PowerTrain.Clutch.DMF.mass

Mass of the DMF spring. Default: 0.325 kg.

PowerTrain.Clutch.DMF.n_mass

Number of mass elements the spring is divided into [1 .. 5]. Default: 3.

PowerTrain.Clutch.DMF.I_spring

Inertia of the spring of the DMF. Default: 0.0085 kgm^2 .

PowerTrain.Clutch.DMF.I_prim

Inertia of the primary side of the DMF. Default: 0.01kgm^2.

PowerTrain.Clutch.DMF.I_sec

Inertia of the secondary side of the DMF. Default: 0.01 kgm^2.

PowerTrain.Clutch.DMF.c

Stiffness of the DMF spring. Default: 3000 Nm/rad.

PowerTrain.Clutch.DMF.k

Damping of the DMF spring. Default: 5 Nms/rad.

PowerTrain.Clutch.DMF.mu

Friction coefficient between the spring mass elements and the primary side. Default: 0.01.

PowerTrain.Clutch.DMF.ra

Outer radius of the DMF spring. Default: 0.125 m.

PowerTrain.Clutch.DMF.rm

Middle radius of the DMF spring. Default: 0.114 m.

PowerTrain.Clutch.DMF.Trq_fric

Constant friction torque between the primary and secondary side. Default: 10 Nm.

Clutch model "Converter"

A hydrodynamic torque converter or Föttinger-Converter is able to reduce rotation speed and translate torque.

The input and output torque $T_{In,Out}$ of the converter is calculated with the speed ratio dependent converter factors k_{In}, k_{Out} and the input speed ω_{In}.

$$T_{In} = k_{In}\omega_{In}^2 \qquad T_{Out} = k_{Out}\omega_{Out}^2 \qquad k_{In}, k_{Out} = f(s) \quad \text{mit} \quad s = \frac{\omega_{Out}}{\omega_{In}}$$

(EQ 181)

The relation between the converter output- and the input-torque is often described with the torque ratio μ.

$$\mu = \frac{T_{Out}}{T_{In}} = \frac{k_{Out}}{k_{In}} \cdot s^2$$

(EQ 182)

So, two characteristics k_{In} (s) and μ (s) respectively k_{Out}(s) are used to represent the transmission behavior of the converter.

Those characteristics use the following parameter names in CarMaker input files:

k_{In}	PowerTrain.Clutch.k_E
μ	PowerTrain.Clutch.mue

Figure 15.54 shows a typical gradient of the converter factor k_{In} over speed ratio n_{Out}/n_{In}:

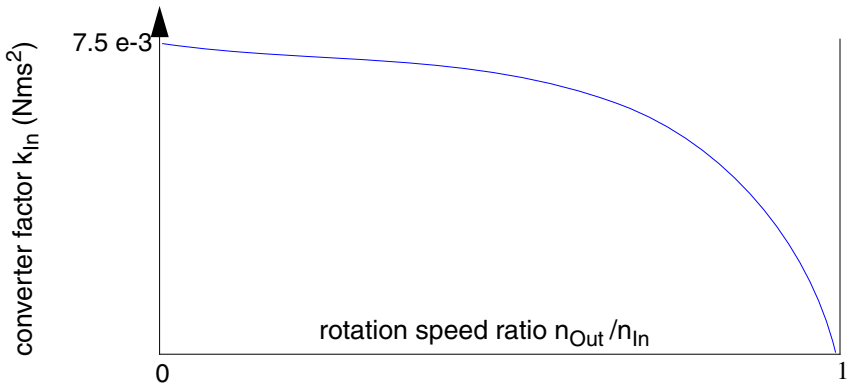


Figure 15.54: converter factor k_{In} as a function of the rotation speed ratio

As depicted in Figure 15.55 the maximum torque ratio (usually 1.9 to 2.5) for the driveway reduces with increasing rotation speed ratios. Above a certain speed ratio the torque ratio remains constantly shortly below 1 (because of losses). This case is called clutch mode.

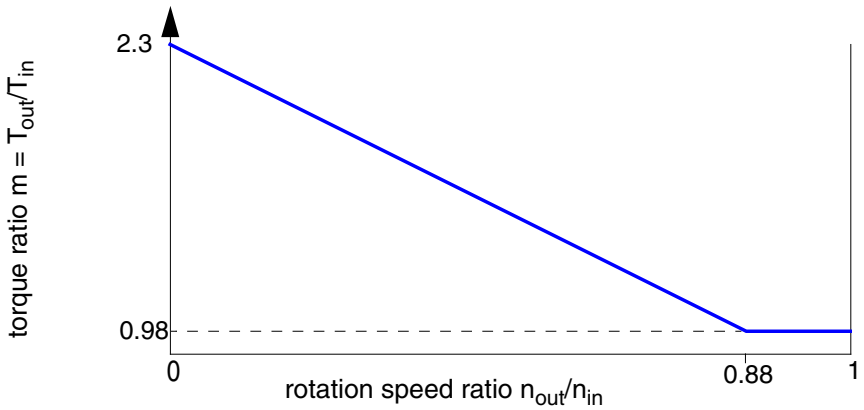


Figure 15.55: torque ratio as a function of the rotation speed ratio

Parameter Converter

PowerTrain.Clutch.Adjust = *value*

For adaption of converter characteristics. This factor is multiplied with `PowerTrain.Clutch.k_E`.

Example `PowerTrain.Clutch.Adjust = 1.0`

PowerTrain.Clutch.k_E: *Table*

Characteristic for the input torque conversion factor. The conversion factor k_E includes the

Syntax Infofile table mapping with 2 columns
<nout/nin> <k_E>

Example PowerTrain.Clutch.`k_E`:

0.0	0.0075
0.2	0.00717
...	...
0.9	0.00296
1.0	0.0

multiplication with the outer diameter D^5.
The rotation ratio 'nout/nin' is limited in the model to values between 0 and 1.

PowerTrain.Clutch.mue : *Table*

Torque ratio characteristic.

Syntax Infofile table mapping with 2 columns
<nout/nin> <mue>

Example PowerTrain.Clutch.`mue`:

0.0	2.1
0.8	0.98
0.9	0.98
1.0	0.98

The rotation ratio 'nout/nin' is limited in the model to values between 0 and 1.

PowerTrain.Clutch.LockUp.c = *value*
PowerTrain.Clutch.LockUp.d = *value*

Sets the spring and damping constants of the internal lockup clutch.
Default: 1000 Nm/rad and 50 Nms/rad.

PowerTrain.Clutch.LockUp.DisconnectPos = *value*

At this position the lockup clutch starts transferring torque when closing. Default: 0.9 [0 .. 1].

Clutch model "DVA"

This model is *not* a conventional clutch model. The transmissible torque is *not* determined by the rotations speed difference of the clutches input and output shaft and the pedal actuation.

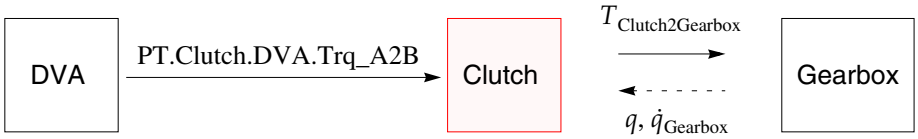


Figure 15.56: Clutch Model “DVA”

The user can specify the clutch torque $T_{\text{Clutch2Gearbox}}$ by modifying the DVA variable *PT.Clutch.DVA.Trq_A2B*.
This model decouples the engine model since only the user specified torque is transferred to the gearbox input shaft.

PowerTrain.Clutch.ClKind = *KindStr*

Optional. Specifies the clutch type of the DVA model. Following kinds are supported:

- *Friction*: Friction clutch (default)
- *Converter*: Converter clutch

Clutch model "Closed"

This model is *not* a conventional clutch model. It replaces the clutch by a rigid shaft, so that torque and rotation angle/speed are equal at the input and output of this model. This model has *no* model specific parameters.

Clutch model "Open"

This model is *not* a conventional clutch model. The input and output shafts are permanently disconnected. This model has *no* model specific parameters.

Clutch model "User C-code / Simulink Plug-in / FMU"

Following parameters are required for the initialization of the output variables in the interface struct *tPTClutchCfgIF*:

PowerTrain.Clutch.ClKind = *ClKindStr*

Optional. Specifies the clutch kind of the user model. Following clutch kinds are supported:

- *Closed*: Permanently closed clutch
- *Open*: Permanently open clutch
- *Friction*: Friction clutch
- *Converter*: Converter clutch

15.4.5 Gearbox

Overview

The primary task of the gearbox subsystem is the transmission of rotation speed and torque from the input to the output shaft according to Figure 15.57.

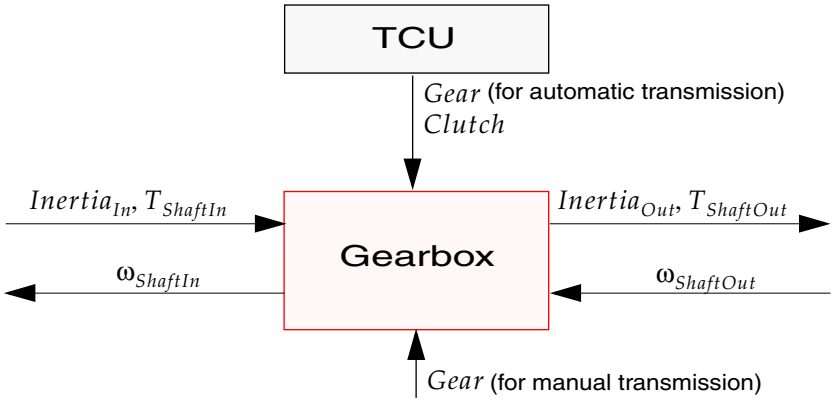


Figure 15.57: Gearbox model

PowerTrain.GearBox.Kind = *KindStr* *VersionId*

Selection of the gearbox subsystem to use. The powertrain components library provides the following gearbox models, some of which have an internal clutch included:

ModelName	KindStr	Description
Manual	Manual	Manual transmission without internal clutch
Automatic	Automatic	Automatic transmission without internal clutch
Auto. Converter	Auto_Conv	Automatic transmission with internal hydraulic converter
Auto. Manual Transmission	Auto_AMT	Automatic transmission with internal friction clutch
Continuously Variable Transmission	CVT	Continuously variable transmission without internal clutch
Dual Clutch Transmission	DCT	Dual clutch transmission with two internal friction clutches
DVA	DVA	Transmission ratio is modified via DVA access
NoGearBox	NoGearBox	Gearbox without gears (fixed ratio=1.0)

Example `PowerTrain.GearBox.Kind = Manual 1`

Some of the gearbox models are not available for all powertrain architectures. Please refer to section 15.7 'Powertrain Models' for more information about which combination of powertrain architecture and gearbox can be handled.

Parameter GearBox

The gearbox models that are provided by the powertrain components library have some parameters in common.

PowerTrain.GearBox.I_in= *value*

Inertia of the gearbox input side. Default: 0.0001 kgm^2.

PowerTrain.GearBox.I_out= *value*

Inertia of the gearbox output side. Default: 0.05 kgm^2.

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tPTGearBoxCfgIF*:

Output Variable	Description
GBKind	Gearbox kind (manual, automatic)
ClKind	Gearbox internal clutch kind (friction, converter)
nFGears	Number of forward gears
iFGear[*gear*]	Ratios of forward gears
nBGears	Number of backward gears
iBGear [*gear*]	Ratios of backward gears

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tPTGearBoxIF*:

Input Variable	Unit	Description
GearNoTrg	-	Gearbox target gear
GearNoTrg_dis	-	Gearbox target gear for disengaged shaft (DCT)
set_ParkBrake	bool	Gearbox set park brake
i_trg	-	Gearbox target gear ratio (CVT)
rot_out	rad	Current gearbox output shaft rotation angle
rotv_out	rad/s	Current gearbox output shaft rotation speed
Trq_in	Nm	Current gearbox input shaft torque
Inert_in	kgm^2	Gearbox input shaft inertia
ClutchIn.Pos Clutch_dis_In.Pos	-	Gearbox internal clutch target position

Output Variable	Unit	Description
GearNo	-	Current gear
GearNo_dis	-	Current gear of disengaged shaft (DCT)
i	-	Current gearbox ratio
i_TrqIn2Out	-	Current gearbox ratio considering loss
rot_in	rad	Current gearbox input shaft rotation angle
rotv_in	rad/s	Current gearbox input shaft rotation speed

Output Variable	Unit	Description
Trq_out	Nm	Current gearbox output shaft torque
Inert_out	kgm^2	Gearbox output shaft inertia
Trq_SuppInert	Nm	Support torque of inertia
ClutchOut.rot_in Clutch_dis_Out.rot_in	rad	Current gearbox clutch input shaft rotation angle
ClutchOut.rotv_in Clutch_dis_Out.rotv_in	rad/s	Current gearbox clutch input shaft rotation speed
ClutchOut.rot_out Clutch_dis_Out.rot_out	rad	Current gearbox clutch output shaft rotation angle
ClutchOut.rotv_out Clutch_dis_Out.rotv_out	rad/s	Current gearbox clutch output shaft rotation speed
ClutchOut.Trq_in Clutch_dis_Out.Trq_in	Nm	Current gearbox input shaft torque
ClutchOut.Trq_out Clutch_dis_Out.Trq_out	Nm	Current gearbox output shaft torque
ClutchOut.i_TrqIn2Out Clutch_dis_Out.i_TrqIn2Out	-	Ratio between gearbox clutch input shaft and output shaft (estimated)

For internal clutch models (clutch model integrated into gearbox model), it is necessary to differentiate the internal interface between the parts of the gearbox model (clutch and gearbox) and the models interface to other powertrain modules. The clutch model's internal interface is not predefined in an interface struct. Its generalized interface to other powertrain modules is defined in the interface struct of the gearbox model (*tPTGearBoxIF*). Please notice, that input and output signals of gearbox and gearbox clutch may be partly redundant (depending on the internal interface). E.g. if the internal clutch is connected directly to the gearbox input shaft, *Trq_in* is equal to *Clutch.Trq_in*.

Gearbox model "Manual"

The selection of the gear ratio with the manual transmission model is done by user requirements or the driver model (IPGDriver). In general, it is combined with an independent friction clutch.

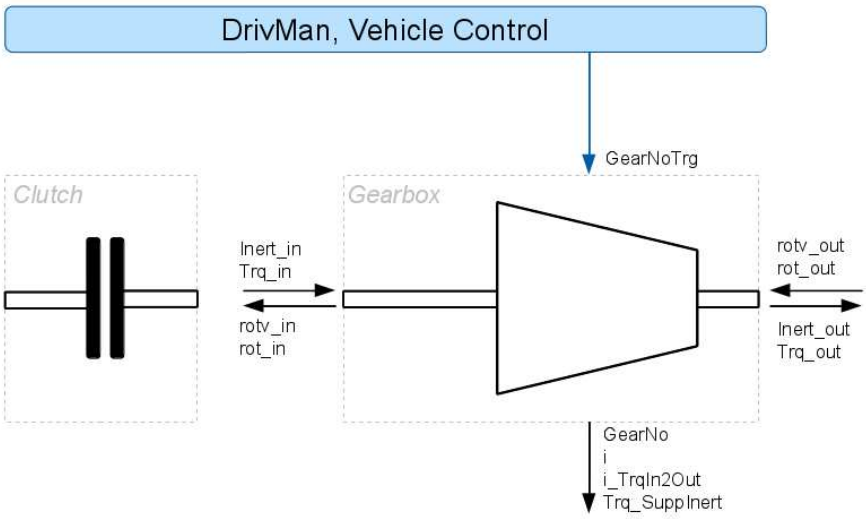


Figure 15.58: Gearbox model "Manual"

The transmission of torque is calculated by this simple approach:

$$T_{Out} = i_G \cdot T_{In} \cdot \eta_G \quad \text{(EQ 183)}$$

The rotation speed of the input shaft is determined accordantly, when transmission ratio is not equal to zero. Otherwise, the input shaft rotation is calculated using the inertias of the gearbox input shaft $I_{InputShaft}$ at its input I_{Input}:

$$\dot{\omega}_{In} = \frac{T_{In}}{I_{Input} + I_{InputShaft}} \quad \text{(EQ 184)}$$

The change from one gear ratio to another has to proceed in a certain time period because it is impossible to accelerate the transmission input inertia unlimited. This synchronization process is modeled through a constant gear change duration (can be parameterized). The transmission rate is adjusted continuously during this procedure.

The ratio between output and input torque is estimated via

$$i_{TrqIn2Out} = i_G \cdot \eta_G \quad \text{(EQ 185)}$$

Parameter Manual

PowerTrain.GearBox.iForwardGears = *GearRatioList*

State the transmission ratio for every single forward gear.

Example
```
PowerTrain.GearBox.iForwardGears = 3.4 1.9 1.35 1.05 0.8
```

PowerTrain.GearBox.iBackwardGears = *GearRatioList*

State the transmission ratio for every single backward gear.

Example
```
PowerTrain.GearBox.iBackwardGears = -4.0
```

PowerTrain.GearBox.nFit = *value*

Specifies the synchronization time during the gear number is linearly interpolated. Default: 50ms.

PowerTrain.GearBox.EtaForwardGears = *GearEfficiencyList*
PowerTrain.GearBox.EtaBackwardGears = *GearEfficiencyList*

State the efficiency for every forward/backward gear. Values must be between 0...1. Default: 1.0. The last efficiency value in the list is valid for all remaining gears.

Example
```
PowerTrain.GearBox.EtaForwardGears = 0.98 0.97 0.96 0.95 0.9
PowerTrain.GearBox.EtaForwardGears = 0.98 0.97
```

Gearbox model "Automatic"

The selection of the gear ratio with the automatic transmission model is done by TCU. In general, it is combined with an independent converter model.

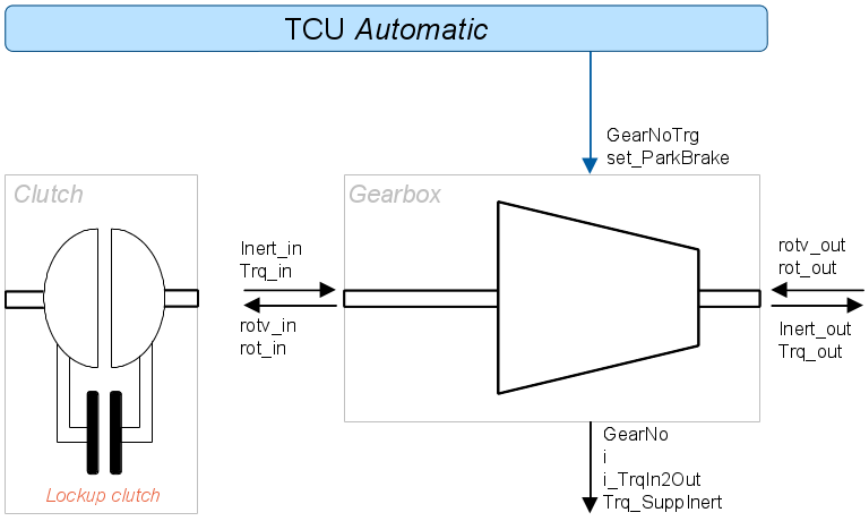


Figure 15.59: Gearbox model "Automatic"

To model the gear transmission, gearbox model *Manual* with all known parameters is used:

Parameter Automatic

PowerTrain.GearBox.iForwardGears =*GearRatioList*
PowerTrain.GearBox.iBackwardGears =*GearRatioList*
PowerTrain.GearBox.EtaForwardGears = *GearEfficiencyList*
PowerTrain.GearBox.EtaBackwardGears = *GearEfficiencyList*

For description details of these parameters see section 'Gearbox model "Manual"'.

Gearbox model "Automatic with Converter"

The gearbox model *Automatic with Converter* is a conventional automatic transmission, which consists of three parts: an internal torque converter with a lockup clutch, an impeller clutch and a gear transmission. The selection of the target gear number and clutch position is done by the TCU.

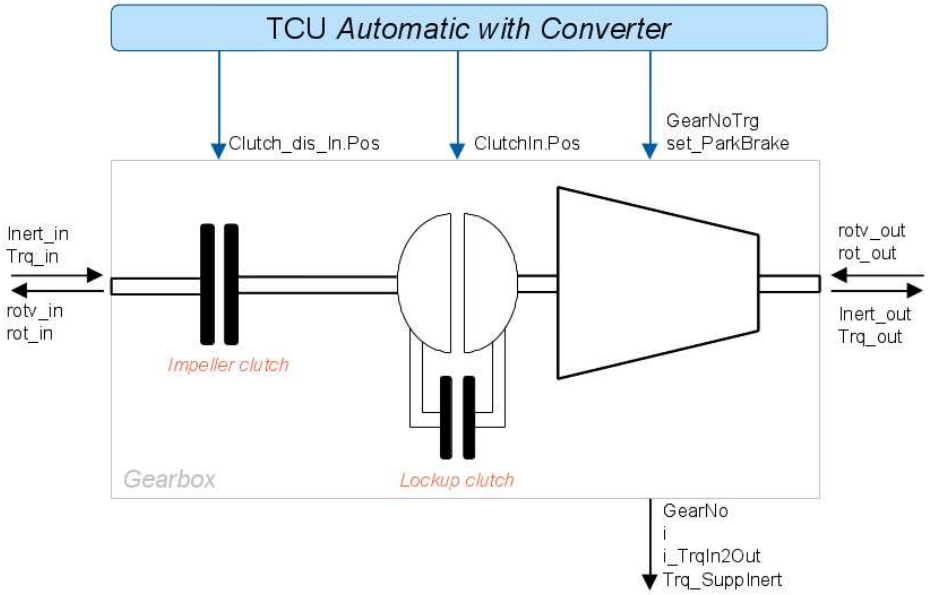


Figure 15.60: Gearbox model "Automatic with Converter"

To model the gear transmission, gearbox model *Manual* with all known parameters is used:

Parameter Automatic with Converter

PowerTrain.GearBox.iForwardGears =*GearRatioList*
PowerTrain.GearBox.iBackwardGears =*GearRatioList*
PowerTrain.GearBox.EtaForwardGears = *GearEfficiencyList*
PowerTrain.GearBox.EtaBackwardGears = *GearEfficiencyList*

For description details of these parameters see section 'Gearbox model "Manual"'.

To model the torque converter with the lockup clutch, the clutch model *Converter* is used as internal clutch model with all known parameters:

PowerTrain.GearBox.Conv.Adjust = *value*
PowerTrain.GearBox.Conv.k_E: *Table*
PowerTrain.GearBox.Conv.mue: *Table*
PowerTrain.GearBox.Conv.LockUp.c = *value*
PowerTrain.GearBox.Conv.LockUp.d = *value*
PowerTrain.GearBox.Conv.LockUp.DisconnectPos = *value*

For description details of these parameters see section 'Clutch model "Converter"'.

The impeller clutch use the same parameters as the lockup clutch but with an other prefix:

PowerTrain.GearBox.Conv.Impeller.c = *value*
PowerTrain.GearBox.Conv.Impeller.d = *value*
PowerTrain.GearBox.Conv.Impeller.DisconnectPos = *value*

Gearbox model "Automated Manual Transmission"

The gearbox model *Automated Manual Transmission* is an automatic transmission, which consists of two parts: an internal friction clutch and a gear transmission. The selection of the gear number and the clutch activation are done by TCU.

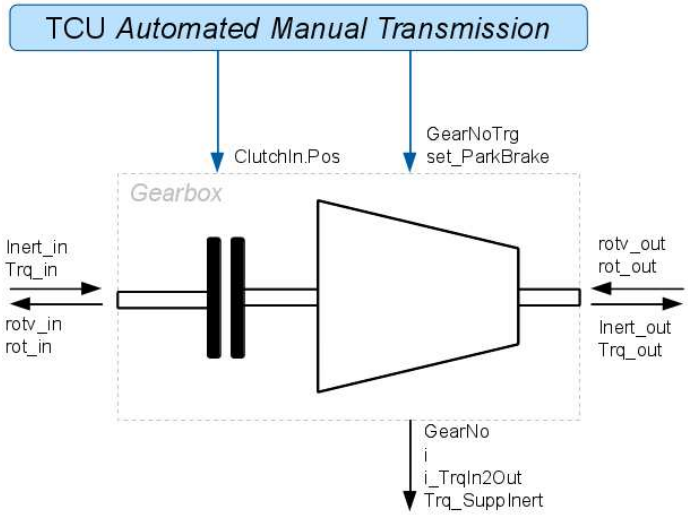


Figure 15.61: Gearbox model "Automated Manual Transmission"

To model the gear transmission, gearbox model *Manual* with all known parameters is used:

Parameter Automated Manual Transmission

PowerTrain.GearBox.iForwardGears =*GearRatioList*
PowerTrain.GearBox.iBackwardGears =*GearRatioList*
PowerTrain.GearBox.EtaForwardGears = *GearEfficiencyList*
PowerTrain.GearBox.EtaBackwardGears = *GearEfficiencyList*

For description details of these parameters see section 'Gearbox model "Manual"'.

To model the friction clutch, the clutch model *Friction* is used as internal clutch model with all known parameters:

PowerTrain.GearBox.Clutch.Trq_max = *value*
PowerTrain.GearBox.Clutch.slope = *value*
PowerTrain.GearBox.Clutch.c = *value*

For description details of these parameters see section 'Clutch model "Friction"'.

It is possible to use only one forward gear and no backward gear ratios for all above listed gearbox models. Then the gearbox is handled as a fix ratio without shifting procedure.
If the gearbox model *Automated Manual Transmission* is used, the internal clutch is always closed and the gearbox does not need to be controlled by the TCU. This is similar to the model *NoGearBox*.
Be aware that the IPG PTControl models only support this functionality for *Automated Manual Transmissions* after an electric motor.

Gearbox model "Continuously Variable Transmission"

The gearbox model *Continuously Variable Transmission (CVT)* is an automatic transmission with a continuously variable transmission ratio. The selection of the transmission ratio is done by the TCU. For backwards driving the highest transmission ratio is used. In general, it is combined with an independent clutch model.

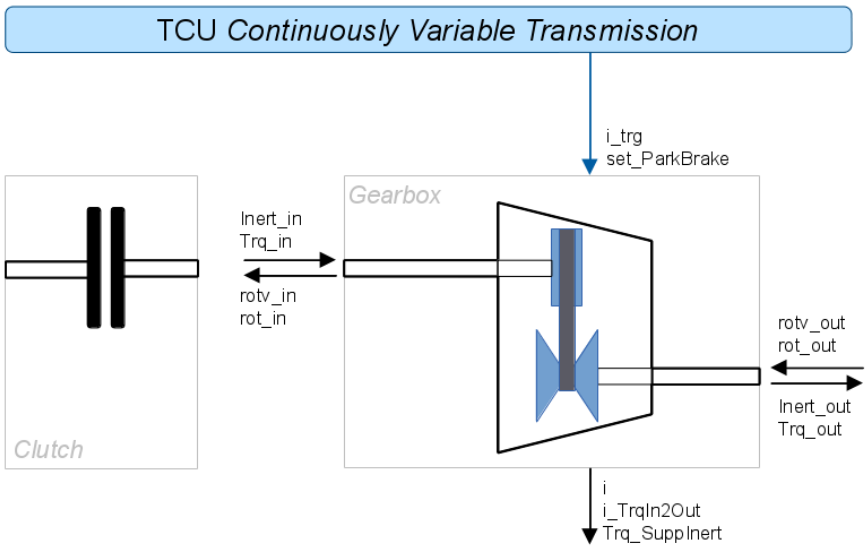


Figure 15.62: Gearbox model "Continuously Variable Transmission"

The transmitted torque is calculated by this simple approach:

$$T_{Out} = i \cdot T_{In} \cdot \eta \qquad \text{(EQ 186)}$$

The rotation speed of the input shaft is determined accordingly.

The ratio cannot change instanteanously.The time the CVT needs to change from the minimal ratio to the maximal ratio is scaled in order to calculate the time needed for smaller transition ratio steps.

The ratio between output and input torque is estimated via

$$i_{TrqIn2Out} = i \cdot \eta \qquad \text{(EQ 187)}$$

Parameter Continuously Variable Transmission

PowerTrain.GearBox.Ratio_min = ***GearRatio***
PowerTrain.GearBox.Ratio_max = ***GearRatio***

State the highest and lowest transmission ratio.

Example

```
PowerTrain.GearBox.maxRatio = 2.5
PowerTrain.GearBox.minRatio = 0.5
```

PowerTrain.GearBox.CVT.tBuildUp = ***value***

Specifies the time the CVT needs to shift from the highest to the lowest transmission ratio. Default: 1s.

PowerTrain.GearBox.Trq_max = *value*

Specifies the maximal torque the CVT is able to transfer.
Default: 400Nm.

PowerTrain.GearBox.EtaMap_rotv: *Table*

One dimensional characteristic mapping for the efficiency depending on the input rotational speed. the input rotational speed is normalized to the maximal engine rotational speed. Efficiency must be between 0 and 1

Syntax Infofile table mapping with 2 columns
<norm. rotation speed> <efficiency>

Example

```
PowerTrain.GearBox.EtaMap_rotv:
0      1.00
0.1    0.997
0.2    0.994
...
1      0.970
```

PowerTrain.GearBox.EtaMap_TrqRatio: *Table*

Two dimensional characteristic mapping for the efficiency depending on the input torque and the current transmission ratio. The input torque is normalized to the maximal transmittable torque of the CVT gearbox. Efficiency must be between 0 and 1

Syntax Infofile table mapping with 3 columns
<norm. torque> <transmission ratio> <efficiency>

Example

```
PowerTrain.GearBox.EtaMap_TrqRatio:
0     0.5   0.85
0     1     0.91
0     1.5   0.90
0     2     0.89
0     2.5   0.87
0.25  0.5   0.88
...
1     2.5   0.93
```

Gearbox model "Dual Clutch Transmission"

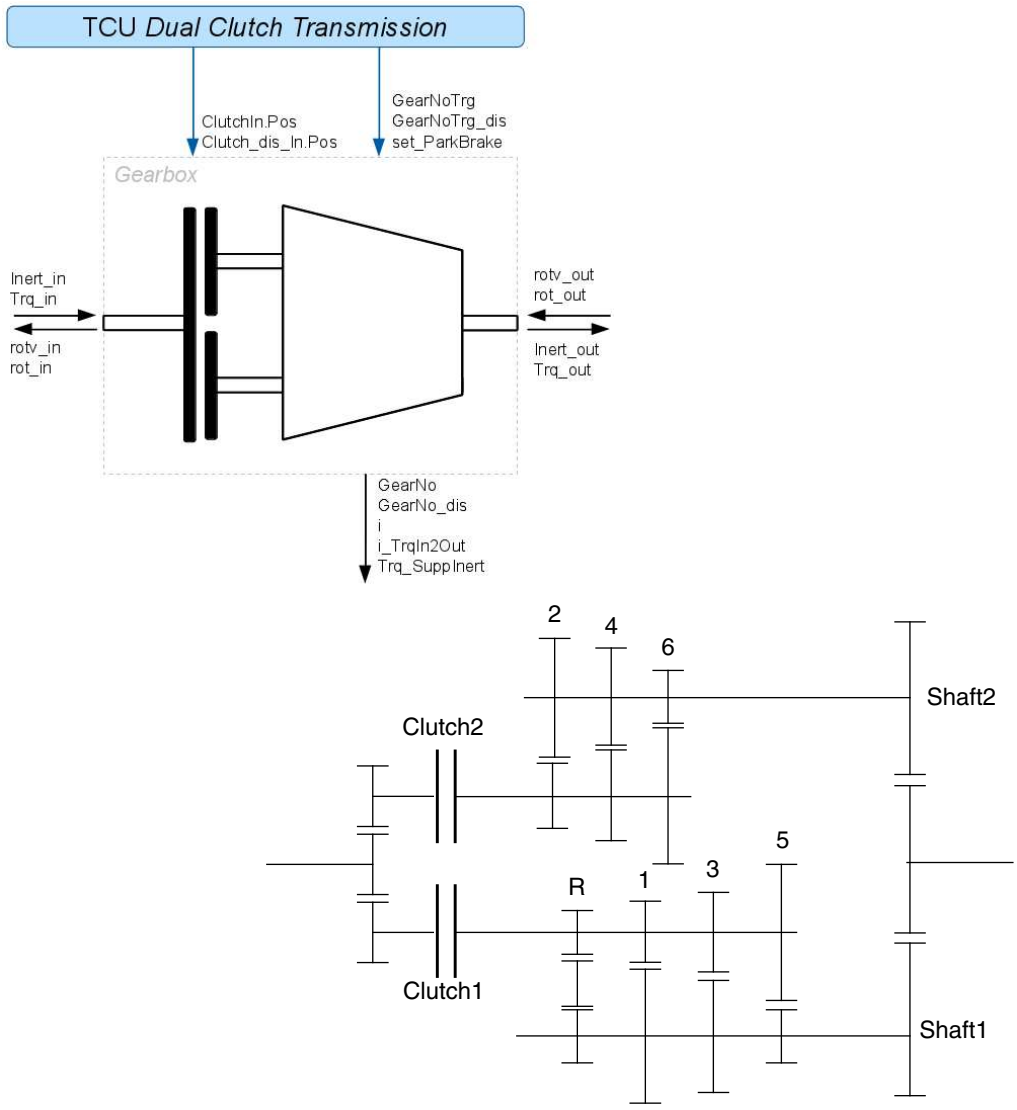


Figure 15.63: Gearbox model "Dual Clutch Tansmission"

The gearbox model *Dual Clutch Transmission* is an automatic transmission that includes two internal friction clutches. The gears are divided onto two separated shafts which are connected to the friction clutches. The selection of the gear and the clutch control are done by the TCU.

This gearbox model supports only one backwards gear. The backwards gear is on the same shaft as the first forward gear. The gear ratios include the complete ratio between input and output of the gearbox, there is no additional ratio modeled between the gear shafts and the output shaft of the gearbox.

The *Dual Clutch Transmission* gearbox can be seen as two separate sub-gearboxes, both consisting of a friction clutch and a gearbox similar to the gearbox model *Manual*. Shaft1 (connected to Clutch1) includes all the odd gears and the backwards gear. Shaft 2 (connected to Clutch2) includes all the even gears. Usually one gear is engaged on each shaft. So, a gear shift can be accomplished by opening one clutch and simultaneously closing the other clutch. An exception is the shift from first gear to backwards gear: As both gears are in the same shaft, both clutches open. The TCU model controls this procedure.

An advantage of the dual clutch transmission is the ability to synchronizes the next gear on the disengaged shaft well in advance.

The parameters used to model the gear transmission are related to the gearbox model *Manual*.

Parameter Dual Clutch Transmission

PowerTrain.GearBox.Shaft1.I_in = ***value***
PowerTrain.GearBox.Shaft2.I_in = ***value***
PowerTrain.GearBox.I_out = ***value***
PowerTrain.GearBox.iForwardGears =***GearRatioList***
PowerTrain.GearBox.iBackwardGears =***GearRatioList***
PowerTrain.GearBox.EtaForwardGears = *****GearEfficiencyList*
PowerTrain.GearBox.EtaBackwardGears = *****GearEfficiencyList*

For description details of these parameters see section 'Gearbox model "Manual"'.

To model the friction effects, two different modes are available. A basic *modelfreeRotOnly* which uses just one torque loss coefficient and applies it to the disengaged, non-synchronized shaft. The more complex model *TrqLossMode* is based on a torque loss map depending on the rotational speed of the shaft.

PowerTrain.GearBox.TrqLossMode = *ModeStr*

State the friction mode for the calculation of the friction effects. Possible values are: freeRotOnly, TrqLossMode.

PowerTrain.GearBox.TrqLossCoeff = *value*

Torque loss coefficient [Nms/rad] to model the behavior of free rotating shafts (non-synchronized) to get to standstill.

PowerTrain.GearBox.TrqLossMap: *Table*

Torque loss map [Nm] dependent on the rotational speed of the rotating shaft

Syntax

Infofile table mapping with 2 columns
<rotational speed> <torque loss>

Example

```
PowerTrain.GearBox.TrqLossMap:
0     0
500   0.005
1000  0.010
1500  0.015
2000  0.020
2500  0.025
4500  0.045
7000  0.07
```

The friction clutches are modeled similar to the Clutch model "Friction" with the following parameter:

PowerTrain.GearBox.Clutch.I_in = ***value***
PowerTrain.GearBox.Clutch1.I_out = ***value***
PowerTrain.GearBox.Clutch2.I_out = ***value***
PowerTrain.GearBox.Clutch.Trq_max = ***value***
PowerTrain.GearBox.Clutch.slope = ***value***
PowerTrain.GearBox.Clutch.c = ***value***

For description details of these parameters see section 'Clutch model "Friction"'.

Parking Lock Torque

All automatic gearbox models from the IPGPowertrain are equipped with a parking lock torque in case if the selector control position *DM.SelectorCtrl* equals *SelectorCtrl_P* (:=-9, park position).

In detail following gearbox models include this functionality:

- *Automatic*
- *Automatic with Converter*
- *Automated Manual Transmission*
- *Continuously Variable Transmission (CVT)*
- *Dual Clutch Transmission (DCT)*

The parking lock torque generates a resistance torque on the gearbox output side (using a linear spring-damper model) against the rolling away in the park situation. Following parameters can be specified optionally:

PowerTrain.GearBox.ParkLock.c

Optional. Stiffness of the parking lock spring. Default: 3000 Nm/rad.

PowerTrain.GearBox.ParkLock.d

Optional. Damping of the parking lock spring. Default: 100 Nms/rad.

Gearbox model "NoGearBox"

This model is only available for electric powertrains.

The gearbox model is a model without gear unit, only with one fixed gear ratio=1.0 and without clutch. This model can be used, if no gearbox is required (for example in electrical powertrain). For this case also a TCU is not required.

Planetary Gearbox model

The planetary gearbox is a special gearbox model with a differing interface, which is not registered as a conventional gearbox model. It is used in the *PowerSplit* hybrid powertrain (for details please refer to section 15.7.7 ’Powertrain model "Power Split Hybrid"’).

Gearbox model "DVA"

This model is *not* a conventional gearbox model. The transmission is *not* determined by the selected gear ratio.

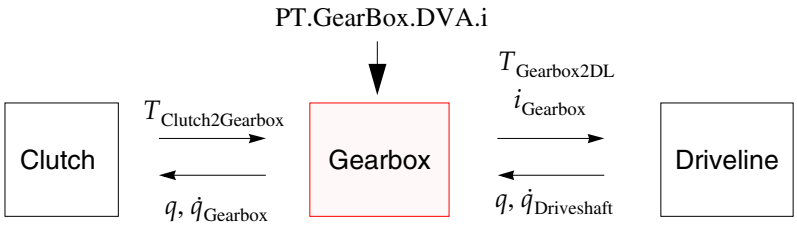


Figure 15.64: Gearbox Model “DVA”

The user can specify a transmission ratio by modifying a DVA variable *PT.GearBox.DVA.i*.

This model is handled by the driver model (IPGDriver) as a manual gearbox type.

Parameter DVA

Following parameters are used for the driver module for a correct switching operation using the gearbox model *DVA*.

PowerTrain.GearBox.nForwardGears = *GearNumber*

State the number of forward gears. Default: 5.

Example `PowerTrain.GearBox.nForwardGears = 7`

PowerTrain.GearBox.nBackwardGears = *GearNumber*

State the number of backward gears. Default: 1.

Example `PowerTrain.GearBox.nBackwardGears = 2`

PowerTrain.GearBox.iForwardGear1 = *GearRatio*

State the transmission ratio of first forward gear. Default: 3.4.
This ratio is used for the driver shifting behavior in the start-up procedure.

Example `PowerTrain.GearBox.iForwardGear1 = 5.1`

PowerTrain.GearBox.iBackwardGear1 = *GearRatio*

State the transmission ratio of first backward gear. Default: -4.0.
This ratio is used for the driver shifting behavior in the start-up procedure.

Example `PowerTrain.GearBox.iBackwardGear1 = -4.5`

Gearbox model "User C-code / Simulink Plug-in / FMU"

The following parameters are required for the initialization of the output variables in the interface struct *tPTGearBoxCfgIF*:

PowerTrain.GearBox.GBKind = *GBKindStr*

Specifies the kind of the user gearbox model. Following kinds are supported:

- *NoGearBox*: No gearbox
- *Manual*: Manual gearbox
- *AutoWithManual*: Automatic gearbox with optional manual gear target (Manumatic)
- *AutoNoManual*: Automatic gearbox without optional manual gear target or continuously variable transmission (CVT)

PowerTrain.GearBox.ClKind = *ClKindStr*

Specifies the clutch kind of the user gearbox model. Following clutch kinds are supported:

- *Closed*: Permanently closed clutch, no clutch
- *Open*: Permanently open clutch
- *Friction*: Friction clutch
- *Converter*: Converter clutch

PowerTrain.GearBox.iForwardGears = *GearRatioList*

State the transmission ratio for every single forward gear.
Only required if manual gearbox or automatic gearbox with manual gear target are used.

Example `PowerTrain.GearBox.iForwardGears = 3.4 1.9 1.35 1.05 0.8`

PowerTrain.GearBox.iBackwardGears = *GearRatioList*

State the transmission ratio for every single backward gear.
Only required if manual gearbox or automatic gearbox with manual gear target are used.

Example `PowerTrain.GearBox.iBackwardGears = -4.0`

15.5 Driveline

Overview

The driveline model transfers the torque from each Drive Source output to the driven wheels. The calculation of wheel speeds and other rotation speeds is done in the computation part of this master module.

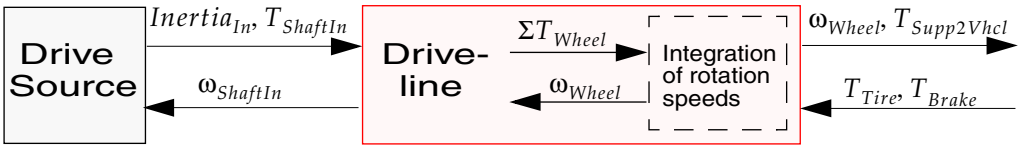


Figure 15.65: Driveline model

PowerTrain.DL.Kind = *KindStr* *VersionId*

Selection of the driveline subsystem to use. The powertrain components library provides the following driveline models:

ModelName	KindStr	Description
Front drive	GenFront	Standard front drive vehicle with rigid drive shafts
Rear drive	GenRear	Standard rear drive vehicle with rigid drive shafts
Front drive, rear axle shiftable	Gen2p2Front	Front driven vehicle with rear axle hanged on with rigid shafts
Rear drive, front axle shiftable	Gen2p2Rear	Rear driven vehicle with front axle hanged on with rigid shafts
All wheel drive	Gen4WD	Four wheel drive vehicle with rigid drive shafts
Universal drive	GenAxle	Flexible axle-wise drive configuration on differential, shafts or wheels

Example

```
PowerTrain.DL.Kind = GenFront 1
PowerTrain.DL.Kind = MyDriveLine 1
```

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tPTDriveLineCfgIF*:

Input Variable	Description
nWheels	Number of vehicle wheels
Wheel_Iyy[*wheel*]	Wheel inertia [kgm^2]

Output Variable	Description
iDiff_mean	Driveline mean differential ratio
DriveSourcePos[*ds*]	Specifies on which position the drive source *ds* is applied

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tPTDriveLineIF*:

Input Variable	Unit	Description
WheelIn[*wh*].Trq_Brake	Nm	Total conventional (e.g: hydraulic) brake torque to the wheel *wh*
WheelIn[*wh*].Trq_T2W	Nm	Tire torque around wheel *wh* spin axis
WheelIn[*wh*].Trq_WhlBearing	Nm	Wheel *wh* bearing torque
DriveIn[*ds*].Trq_in	Nm	Drive source *ds* input torque
DriveIn[*ds*].Inert_in	kgm^2	Drive source *ds* input inertia

Output Variable	Unit	Description
Trq_Supp2Bdy1	Nm	Drive torque support on vehicle body Fr1A
Trq_Supp2Bdy1B	Nm	Drive torque support on vehicle body Fr1B
Trq_Supp2BdyEng	Nm	Drive torque support on engine body FrEng
iDiff_mean		Driveline mean differential ratio (overrides the ratio from the initialization if non zero)
WheelOut[*wh*].rot	rad	Current wheel *wh* rotation angle
WheelOut[*wh*].rotv	rad/s	Current wheel *wh* rotation speed
WheelOut[*wh*].Trq_Drive	Nm	Current drive torque at wheel *wh*
WheelOut[*wh*].Trq_B2W	Nm	Current reduced brake torque at wheel *wh*
WheelOut[*wh*].Trq_Supp2WC	Nm	Drive torque support on wheel *wh* carrier
DriveOut[*ds*].rot_in	rad	Drive source *ds* input rotation angle
DriveOut[*ds*].rotv_in	rad/s	Drive source *ds* input rotation speed
DriveOut[*ds*].i_D2W[*wh*]	-	Drive source *ds* current ratio drive source to wheel *wh* (estimated)

Driveline - Generic

This section describes the available driveline modules provided with the ‘Generic’ library. The models of this library are highly configurable and should meet the needs of the most common configurations.

Basically driveline configurations are distinguished by the number of driven wheels and in addition to this by the number of differential gears used. Figure 15.66 gives an overview of the available configurations:

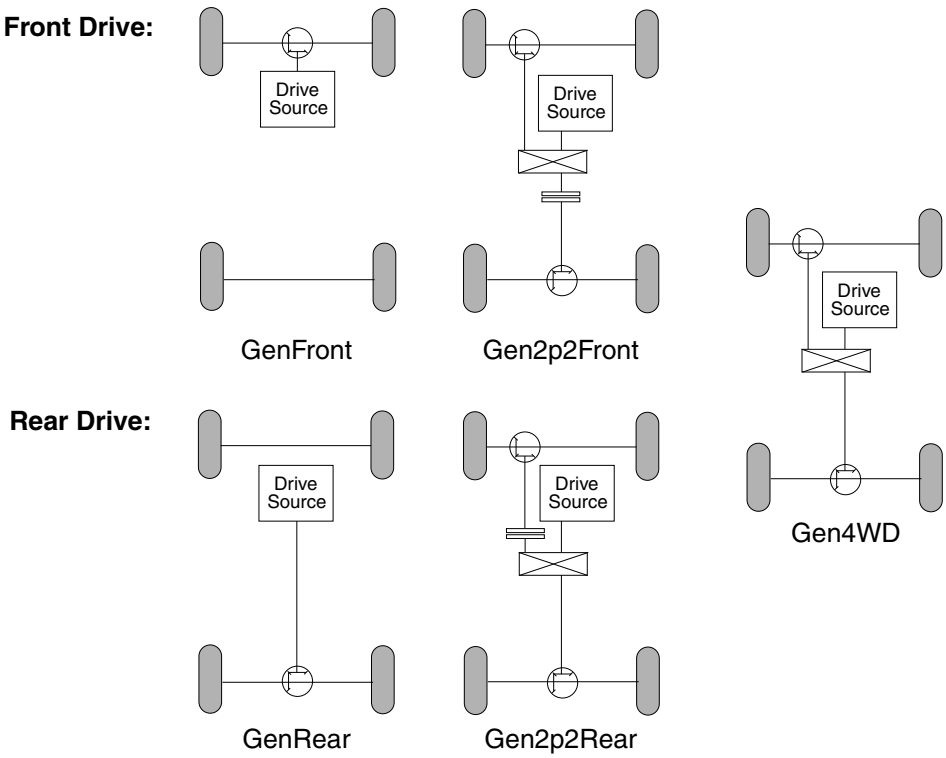


Figure 15.66: Configurations overview of basic driveline configurations

The standard front and rear driveline configurations use one differential gear to distribute the torque from the drive shaft to the wheels (either at the front or at the rear axle). The 4WD configuration uses one differential gear for every axle and a center differential gear for distributing torque to the front and rear axle. The hanged on configurations also have differential gears for front and rear axle but use a coupling for transferring torque to the hanged on axle. This means that with opened center coupling this configuration acts like a standard front or rear drive.

Additional Drive Source in GenFront/ GenRear

Using the driveline configuration ‘GenFront’ or ‘GenRear’ the drive torque coming from the first *Drive Source* (e.g. engine torque) is distributed to the wheels with one differential gear (either front or rear). The user have the possibility to apply additional torque from a second *Drive Source 1* to the undriven wheels (e.g. torque coming from an electric motor).

Figure 15.67 shows two possibilities to apply additional torque: torque to an additional differential gear or directly to the wheels. If the first is used, the additional differential gear must be parametrized as usual.

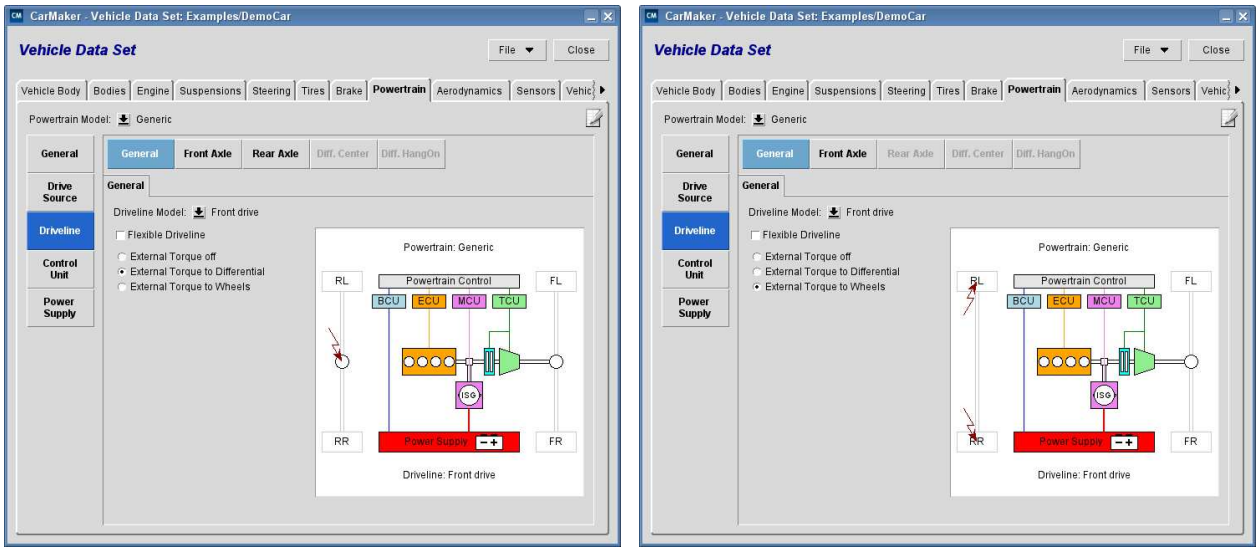


Figure 15.67: Vehicle Data Set: choosing external torque to driveline

Choosing the option “External Torque to Differential“ in the GUI corresponds to the vehicle data file key:

Example `PowerTrain.DL.TrqExt2DiffOn = 1`

The external torque can be manipulated via modification of the DVA quantity *PT.Gen.DL.DriveSrc.1.Trq_ext* or directly in the C-Code using the parameter *PowerTrain.DriveLine.Trq_Ext2DriveSrc[1]*.

The applied external torque to the differential is supported by the vehicle body.

Choosing the option “External Torque to Wheels“ in the GUI corresponds to the vehicle data file key:

Example `PowerTrain.DL.TrqExt2WheelOn = 1`

The external torque can be manipulated in same manner via modification of the DVA quantities *PT.DL.DriveSrc.1.Trq_ext* / *PT.DL.DriveSrc.2.Trq_ext* or directly in the C-Code using the parameter *PowerTrain.DriveLine.Trq_Ext2DriveSrc[1]* / *PowerTrain.DriveLine.Trq_Ext2DriveSrc[2]*.

The applied external torques to the wheels are supported by the wheel carriers.

Model “Generic” Differential Parameters

Parameter Differential Models “Front” or “Rear”

Depending on the driveline configuration a differential gear is used either for front or rear or both axles.

PowerTrain.DL.FDiff.I_in = ***value***
PowerTrain.DL.RDiff.I_in = ***value***

Inertia of the differential input shaft.

PowerTrain.DL.FDiff.I_out = ***value***
PowerTrain.DL.RDiff.I_out = ***value***

Inertia of the differential output shaft.

PowerTrain.DL.FDiff.I_Cage = ***value***
PowerTrain.DL.RDiff.I_Cage = ***value***

Inertia of the differential cage. Default: 0.

PowerTrain.DL.FDiff.i = ***value***
PowerTrain.DL.RDiff.i = ***value***

Transmission ratio from differential input shaft to cage.

Example
```
PowerTrain.DL.FDiff.i = 3.5
```

PowerTrain.DL.FDiff.Eta = ***EfficiencyValue***
PowerTrain.DL.RDiff.Eta = ***EfficiencyValue***

State the efficiency for front/rear differential. Values must be between 0...1. Default: 1.0.

Example
```
PowerTrain.DL.FDiff.Eta = 0.9
PowerTrain.DL.RDiff.Eta = 0.95
```

PowerTrain.DL.FDiff.Cpl.Kind = ***KindStr***
PowerTrain.DL.RDiff.Cpl.Kind = ***KindStr***
PowerTrain.DL.FDiff.Cpl.Mounting = ***MountPos***
PowerTrain.DL.RDiff.Cpl.Mounting = ***MountPos***
PowerTrain.DL.FDiff.Cpl.k
PowerTrain.DL.RDiff.Cpl.k

For details of the coupling parameters please refer to section 15.5.1 ’Coupling’.

Parameter Differential Model “Central”

A center differential is *only* available for Model “All wheel drive”.

PowerTrain.DL.CDiff.I_in = *value*

Inertia of the input shaft.

PowerTrain.DL.CDiff.I_out_front = *value*

Inertia of the front output shaft.

PowerTrain.DL.CDiff.I_out_rear = *value*

Inertia of the rear output shaft.

PowerTrain.DL.CDiff.I_Cage = *value*

Inertia of the differential cage. Default: 0.

PowerTrain.DL.CDiff.i_in2cent = *GearRatioList*

State the transmission ratio from input shaft to cage of the center differential for every gear. The center differential supports until four gears.

Example `PowerTrain.DL.CDiff.i_in2cent = 1.0 0.9 0.8`

PowerTrain.DL.CDiff.TrqRatio_front = *value*

Selects the torque distribution between front and rear axle. This parameter specifies the percentage of the input torque which is transferred to the front axle (the remaining torque goes to the rear axle). The input range is from 0..1 excluding both extreme values.

Example `PowerTrain.DL.CDiff.TrqRatio_front = 0.7`

This means that 70% of the input torque is transferred to the front axle and 30 % to the rear axle.

PowerTrain.DL.CDiff.Eta = *GearEfficiencyList*

State the efficiency for every central differential gear. Value must be between 0...1. Default: 1.0 for each gear. The center differential supports until four gears.

Example `PowerTrain.DL.CDiff.Eta = 0.97 0.95`

PowerTrain.DL.CDiff.GearNo = *value*

Specifies the initial center differential gear number if there exists more than one. Default: 1. The current gear number can be manipulated via Direct Variable Access (DVA) on the quantity *PT.Gen.DL.CDiff.GearNo*.

PowerTrain.DL.CDiff.Cpl.Kind = *KindStr*
PowerTrain.DL.CDiff.Cpl.Mounting = *MountPos*
PowerTrain.DL.CDiff.Cpl.k = *value*

For details of the coupling parameters please refer to section 15.5.1 'Coupling'.

Parameter Differential Models "Gen2p2Front" or "Gen2p2Rear"

One axle is driven, the other one is hanged on. It is parametrized with basic parameters, extended by a hang on coupling.

PowerTrain.DL.HangOn.I_in = *value*

Inertia of the input shaft.

PowerTrain.DL.HangOn.I_out = *value*

Inertia of the output shaft.

PowerTrain.DL.HangOn.i = *value*

Transmission ratio between gearbox and hangon coupling.

PowerTrain.DL.HangOn.rotv_open = *value*

Only for configuration Gen2p2Front! Optional. With this parameter it is possible to implement a disconnect unit for the rear axle. When the vehicle is dragged and the rotation speed is above the specified value the disconnect unit applies. Unit: [rpm].
Default: 1e38 (this means *no* disconnect unit is installed).

Example `PowerTrain.DL.HangOn.rotv_open = 100`

PowerTrain.DL.HangOn.Cpl.Kind = *KindStr*
PowerTrain.DL.HangOn.Cpl.k = *value*

For details of the coupling parameters please refer to section 15.5.1 'Coupling'.

Driveline model "Universal drive"

The driveline model 'Universal drive' (GenAxle) allows to define flexibly on each axle if a drive source acts on differential, wheels or shafts. This is comfortable for powertrains with more than one drive source (e.g. engine in combination with electric motor or two electric motors).

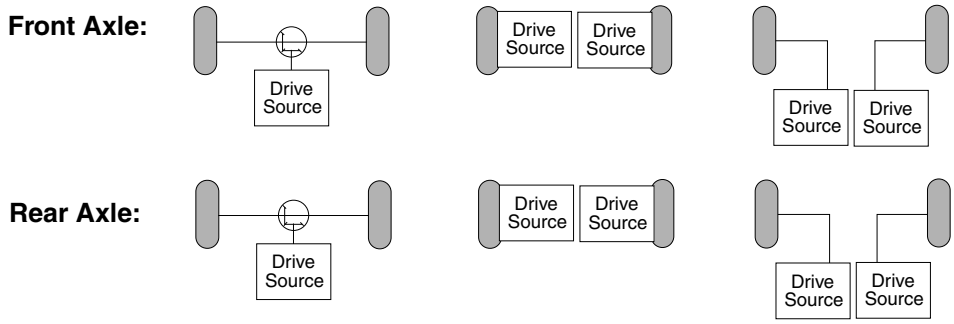


Figure 15.68: Configurations overview of 'GenAxle' driveline

PowerTrain.DL.FAxle.DrvSrc = Diff *DriveSourceId*
PowerTrain.DL.FAxle.DrvSrc = Shaft *DriveSourceId DriveSourceId*
PowerTrain.DL.FAxle.DrvSrc = Wheel *DriveSourceId DriveSourceId*
PowerTrain.DL.RAxle.DrvSrc = Diff *DriveSourceId*
PowerTrain.DL.RAxle.DrvSrc = Shaft *DriveSourceId DriveSourceId*
PowerTrain.DL.RAxle.DrvSrc = Wheel *DriveSourceId DriveSourceId*

Specifies if the axle is driven by input torque to differential, shafts or wheels. Using drive torque to differential only one drive source is required. Using drive torque to shafts or wheels two different drive sources are required. These could come from the powertrain model (engine or electric motor as drive source) or also as external torque via DVA.

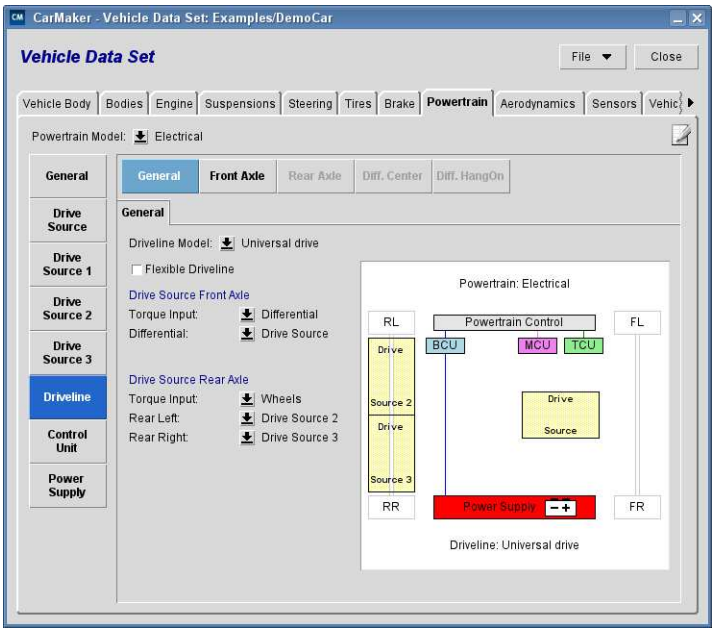


Figure 15.69: Universal drive

Flexible Shafts

Each driveline model from the ‘Generic’ library can be simulated with rigid (default) or with flexible half and drive shafts. Each shaft has an additional degree of freedom for the shaft rotation, realized by the implementation of a linear spring-damper element in the shafts.

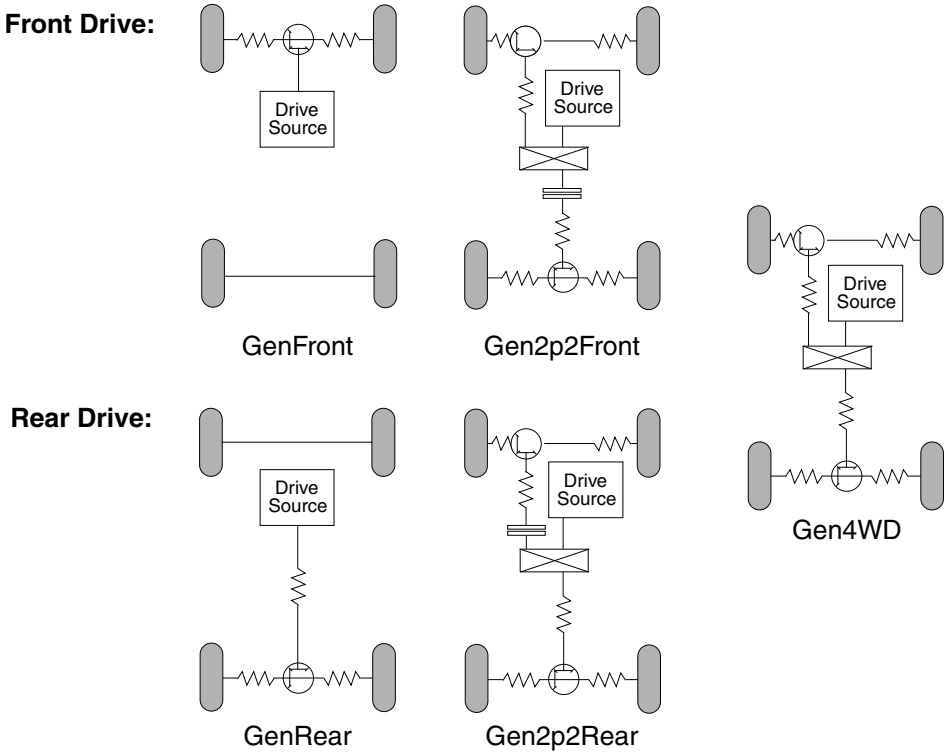


Figure 15.70: Configurations overview with flexible shafts

With the driveline model ’GenAxle’ it is possible in same way to use a flexible half shaft if the *Drive Source* is connected to a differential or to the shafts:

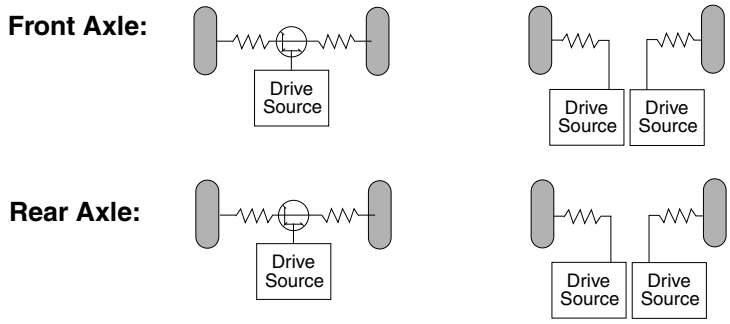


Figure 15.71: Configurations overview of ’GenAxle’ driveline

PowerTrain.DL.FlexShaft = *bool*

Activation of flexible half and drive shafts. Default: 0.

PowerTrain.DL.FDiff.DriveShaft.c = ***Stiffness***
PowerTrain.DL.RDiff.DriveShaft.c = ***Stiffness***

Rotational stiffness coefficient of drive shaft. Unit: [Nm/rad]. Default: 15e3. Depending on the driveline configuration drive shafts are included in the model or not (see Figure 15.70).

PowerTrain.DL.FDiff.DriveShaft.k = ***Damping***
PowerTrain.DL.RDiff.DriveShaft.k = ***Damping***

Rotational damping coefficient of drive shaft. Unit: [Nms/rad]. Default: 15. Depending on the driveline configuration drive shafts are included in the model or not (see Figure 15.70).

PowerTrain.DL.FDiff.HalfShaftL.c = ***Stiffness***
PowerTrain.DL.FDiff.HalfShaftR.c = ***Stiffness***
PowerTrain.DL.RDiff.HalfShaftL.c = ***Stiffness***
PowerTrain.DL.RDiff.HalfShaftR.c = ***Stiffness***

Rotational stiffness coefficient of left/right half shaft. Unit: [Nm/rad]. Default: 10e3.

PowerTrain.DL.FDiff.HalfShaftL.k = ***Damping***
PowerTrain.DL.FDiff.HalfShaftR.k = ***Damping***
PowerTrain.DL.RDiff.HalfShaftL.k = ***Damping***
PowerTrain.DL.RDiff.HalfShaftR.k = ***Damping***

Rotational damping coefficient of left/right half shaft. Unit: [Nms/rad]. Default: 10.

With the activated functionality for flexible shafts, the powertrain oversampling rate *PowerTrain.OSRate* is observed und adapted dynamically during the simulation, if an numerical oscillation occurs. Since the simulation time step for the powertrain can be changed during the simulation, the flexible shafts feature doesn't work in combination with any *Simulink Plugin-In* powertrain modules.

Driveline model "User C-code / Simulink Plug-in / FMU"

The following parameters are required for the initialization of the output variables in the interface struct *tPTDriveLineCfgIF*:

PowerTrain.DL.iDiff_mean =*value*

Specifies the driveline mean differential ratio. Default: 3.0.

PowerTrain.DL.nDriveSource = *value*

Specifies the number of possible drive sources to the driveline. Default: 1.

PowerTrain.DL.DriveSourcePos<ds> = *PosStr*

Specifies on which position the drive source *ds* is applied. Possible positions are:

- *NoPosition*: No position specified
- *Center*: Center differential
- *Front*: Front differential
- *Front2*: Second front differential
- *Rear*: Rear differential
- *Rear2*: Second rear differential
- *FL*: Wheel FL
- *FR*: Wheel FR
- *RL*: Wheel RL
- *RR*: Wheel RR
- *RL2*: Wheel RL2
- *RR2*: Wheel RR2
- *FL2*: Wheel FL2
- *FR2*: Wheel FR2

15.5.1 Coupling

PowerTrain.DL.FDiff.Cpl.Kind = ***KindStr VersionId***
PowerTrain.DL.RDiff.Cpl.Kind = ***KindStr VersionId***
PowerTrain.DL.CDiff.Cpl.Kind = ***KindStr VersionId***
PowerTrain.DL.HangOn.Cpl.Kind = ***KindStr VersionId***

Selection of a coupling model for a differential or hang-on. Default: no coupling is used. The powertrain components library provides the following coupling models:

ModelName	KindStr	Diff	HangOn	Description
Viscoelastic	Visco	x	x	Viscoelastic coupling
Torque Sensing	TrqSensing	x		Torque sensing coupling
Torque Vectoring	TrqVec	x		Variable torque distribution by DVA
Locked	Locked	x	x	Locked rigid connection
Locked by DVA	DVA_Locked	x	x	Variable locking torque by DVA

Example `PowerTrain.DL.FDiff.Cpl.Kind = TrqVec 1`

Parameter Coupling

The coupling models that are provided by the powertrain components library have some parameters in common.

PowerTrain.DL.FDiff.Cpl.k = ***value***
PowerTrain.DL.RDiff.Cpl.k = ***value***
PowerTrain.DL.CDiff.Cpl.k = ***value***
PowerTrain.DL.HangOn.Cpl.k = ***value***

Friction coefficient used for numerical stability when switching from stick to slip state. Unit: Nms/rad. Default: 10.

PowerTrain.DL.FDiff.Cpl.Mounting = ***MountPos***
PowerTrain.DL.RDiff.Cpl.Mounting = ***MountPos***
PowerTrain.DL.CDiff.Cpl.Mounting = ***MountPos***

Select mounting position.

Possible values for *FDiff*, *RDiff* are: `left2right` (default), `in2left`, `in2right`.
Possible values for *CDiff* are: `front2rear` (default), `in2rear`, `in2front`.

There are several possibilities to add different types of couplings between different shafts of the differential gears in the driveline:

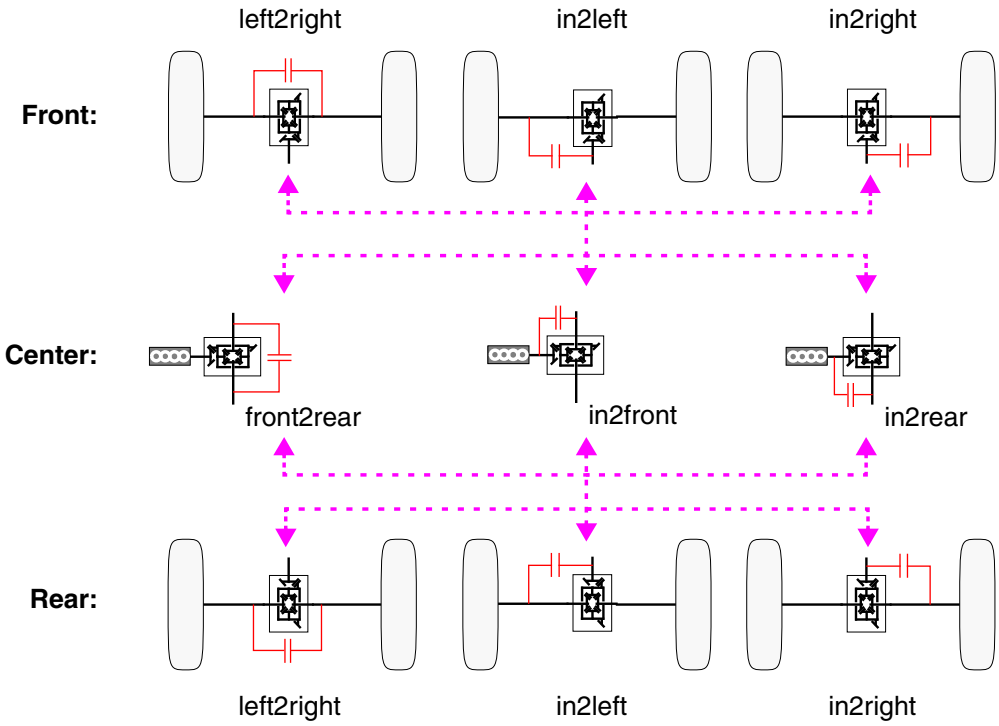


Figure 15.72: Configuration possibilities for driveline couplings

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tPTGenCouplingCfgIF*:

Input Variable	Description
CplPos	Coupling position (front, rear, center differential, hang-on)

Output Variable	Description
CplType	Coupling type: 0: Unknown 1: Not lockable 2: Lockable 3: Fully active

Following coupling type is used in the provided coupling models:

Coupling Model	CplType
Viscoelastic	NotLockable
Torque Sensing	Lockable
Locked	Lockable
Locked by DVA	Lockable
Torque Vectoring	FullyActive

With the keys *NotLockable* and *Lockable* the locking torque is directed towards the shaft with the smaller rotation speed. With the key *FullyActive*, the direction of the locking torque depends on the torque distribution value *TrqRatio*.

With the key *Lockable* the coupling model manages the stick-slip behavior of the coupling torque. For the slip case the effective coupling torque corresponds to the calculated locking torque $T_{LockEff} = T_{Lock}$. For the stick case the effective coupling torque corresponds to a reduced locking torque, depending on the difference of shaft reaction torques $T_{LockEff} = f(T_{Left} - T_{Right})$.

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tPTGenCouplingIF*:

Input Variable	Unit	Description
Trq_Ext2Cage	Nm	Input torque to differential cage
drotv_A2B	rad/s	Rotation speed difference A to B (left to right or front to rear)

Output Variable	Unit	Description
Trq_A2B	Nm	Coupling torque A to B (left to right or front to rear)

Coupling model "Viscoelastic"

As a matter of principle a visco coupling is only able to transfer torque if there is a rotation speed difference between both sides of the coupling. This is because a fluid is used for torque transmission like in a hydrodynamic torque converter (Föttinger-Coupling). This means there is no state where a visco coupling is sticking, there has to be slip for torque transmission. For simulation purposes a torque characteristic as a function of rotating speed difference is used. The coupling torque is calculated by:

$$T = f(\Delta\dot{\omega}) \qquad \text{(EQ 188)}$$

A typical characteristic may look like in Figure 15.73

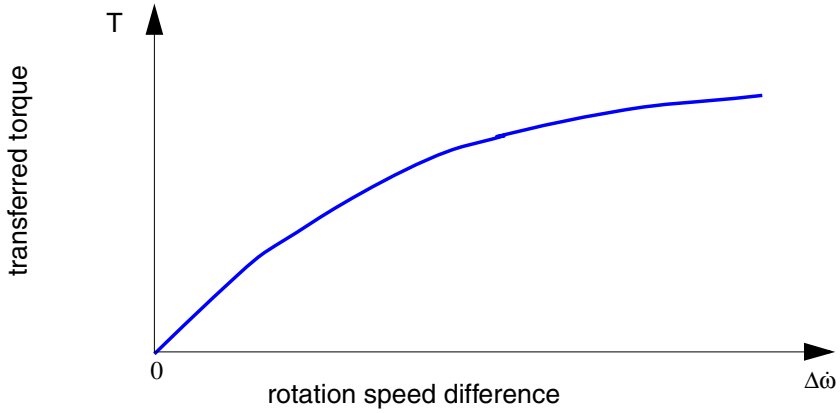


Figure 15.73: Typical characteristic for a visco coupling

PowerTrain.DL.FDiff.Cpl.Trq_Amplify = ***value***
PowerTrain.DL.RDiff.Cpl.Trq_Amplify = ***value***
PowerTrain.DL.CDiff.Cpl.Trq_Amplify = ***value***
PowerTrain.DL.HangOn.Cpl.Trq_Amplify = ***value***

Amplifies the determined value by a given factor. Default: 1.0.

PowerTrain.DL.FDiff.Cpl.Trq_drotv = ***value***
PowerTrain.DL.RDiff.Cpl.Trq_drotv = ***value***
PowerTrain.DL.CDiff.Cpl.Trq_drotv = ***value***
PowerTrain.DL.HangOn.Cpl.Trq_drotv = ***value***

Characteristic for coupling torque as a function of the rotation speed difference: $T = f(\Delta\dot{\omega})$.!

Syntax Infofile table mapping with 2 columns
<rotation velocity difference> <torque>

Coupling model "Torque Sensing"

The coupling torque (locking torque) depends on the torque difference between the two coupled shafts.

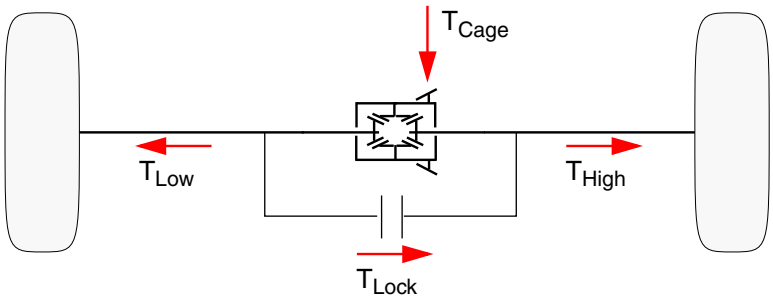


Figure 15.74: Principle of torque sensing coupling

With torque sensing couplings there are two common characteristic values. The *torque bias* and *locking ratio* are defined as:

$$TB = \frac{T_{High}}{T_{Low}} \qquad LR = \frac{T_{Lock}}{T_{Cage}} \tag{EQ 189}$$

If $T_{Lock} = 0$ then $T_{High} = T_{Low} = \frac{1}{2}T_{Cage}$ applies for a standard differential gear. If T_{Lock} exists the torques calculate to:

$$T_{Low} = \frac{1}{2}(T_{Cage} - T_{Lock})$$
$$T_{High} = \frac{1}{2}(T_{Cage} + T_{Lock}) \tag{EQ 190}$$

With (EQ 189), (EQ 189) and (EQ 190) a relationship between torque bias and locking ratio can be determined:

$$LR = \frac{TB - 1}{TB + 1} \tag{EQ 191}$$

The CarMaker implementation of a torque sensing coupling uses a torque bias value as input for each the driven ($T_{Cage} \geq 0$) and undriven ($T_{Cage} < 0$) case.

Following parameters use <xDiff> for *FDiff*, *RDiff* and *CDiff*:

PowerTrain.DL.<xDiff>.Cpl.TrqBias_Driven = *value*

Specify torque bias value for drive case driven. Value has to be >= 1.

Example `PowerTrain.DL.FDiff.Cpl.TrqBias_Driven = 1.3`

PowerTrain.DL.<xDiff>.Cpl.TrqBias_Dragged = *value*

Specify torque bias value for drive case dragged. Value has to be >= 1.

Example `PowerTrain.DL.FDiff.Cpl.TrqBias_Dragged = 1.2`

The coupling model "Torque Sensing" uses for a better stick/slip transition a higher locking ratio *LR_max* for very small speed difference (stick case). With higher speed difference the locking ratio takes down to the normal locking ratio value.

PowerTrain.DL.<xDiff>.Cpl.LR_max = *value*

Optional: Specify the maximum locking ratio for zero speed difference (stick case). Default: 0.8.

PowerTrain.DL.<xDiff>.Cpl.drotv_ref = *value*

Optional: Specify the reference speed difference for the transition from normal value to the maximum locking ratio value. Default: 50. Unit: [rpm].

Coupling model "Torque Sensing" considering speed difference

The coupling model "Torque Sensing" can also work in an extended mode considering the speed difference between the both left/right or front/rear shafts.

Following parameters use <xDiff> for *FDiff*, *RDiff* and *CDiff*:

PowerTrain.DL.<xDiff>.Cpl.WithSlip =*bool*

Specify the mode with speed difference. Default: 0.

Example `PowerTrain.DL.FDiff.Cpl.WithSlip = 1`

If the mode is activated, following parameters are used:

PowerTrain.DL.<xDiff>.Cpl.TrqBias_Driven_wSlip = *value1 value2*

Specify torque bias value for drive case driven. Values has to be >= 1.
Value1 is considered for the case if the left (front) shaft speed is greater than the right (rear) speed, otherwise *Value2* is used.

Example `PowerTrain.DL.FDiff.Cpl.TrqBias_Driven_wSlip = 3.5 2.5`

PowerTrain.DL.<xDiff>.Cpl.TrqBias_Dragged_wSlip = *value1 value2*

Specify torque bias value for drive case dragged. Values has to be >= 1.
Value1 is considered for the case if the left (front) shaft speed is greater than the right (rear) speed, otherwise *Value2* is used.

Example `PowerTrain.DL.FDiff.Cpl.TrqBias_Dragged_wSlip = 2.4 3.3`

Coupling model "Torque Vectoring"

With this coupling the locking torque depends on the torque distribution (torque ratio):

$$T_{Lock} = T_{In}(1 - 2 \cdot TrqRatio) \quad \textbf{(EQ 192)}$$

The torque distribution $TrqRatio$ indicates how much torque is directed to the left (respectively to the front) outshaft. With this coupling kind, the locking torque can be negative.

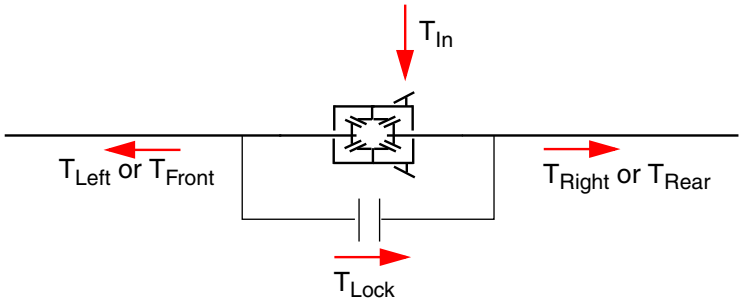


Figure 15.76: Principle of torque vectoring coupling

Using torque vectoring, the configuration left2right (respectively front2rear) should be chosen.

The torque distribution can be manipulated via modification of correspond DVA quantities (See section 24.12.11 'Driveline') or directly in the C-Code using the parameters:

- PTGenCpl_TrqVec.Front.TrqRatio
- PTGenCpl_TrqVec.Center.TrqRatio
- PTGenCpl_TrqVec.Rear.TrqRatio

PowerTrain.DL.FDiff.Cpl.TrqRatio_left
PowerTrain.DL.RDiff.Cpl.TrqRatio_left

Selects the initial torque distribution between left and right axle using torque vectoring. This parameter specifies how the percentage of the input torque which is transferred to the left axle (the remaining torque goes to the right axle). The input range is from 0..1. Default: 0.5

Example `PowerTrain.DL.FDiff.Cpl.TrqRatio_left = 0.7`

This means that 70% of the input torque is transferred to the left axle and 30 % to the right axle.

PowerTrain.DL.CDiff.TrqRatio_front

Selects the initial torque distribution between front and rear axle. This parameter specifies how the percentage of the input torque which is transferred to the front axle (the remaining torque goes to the rear axle). The input range is from 0..1 excluding both extreme values.

Example `PowerTrain.DL.CDiff.TrqRatio_front = 0.7`

This means that 70% of the input torque is transferred to the front axle and 30 % to the rear axle.

Coupling model "Locked"

This is not a real coupling. It acts like a rigid connection between input and output shaft. It is useful for simulating a locked differential or testing purposes. No matter which mount position is chosen the differential is lock in either case. There are no parameters needed for this coupling.

Coupling model "Locked by DVA"

With this coupling the locking torque for the configured couplings can be given via modification of DVA quantities, described in section 24.12.11 'Driveline'.

Coupling model "User C-code / Simulink Plug-in / FMU"

The following parameters are required for the initialization of the output variables in the interface struct *tPTGenCouplingCfgIF*:

PowerTrain.DL.<Diff>.Cpl.CplType = *CplTypeStr*

Specifies the coupling type. Following coupling types are known:

- *Unknown*: Unknown, not specified coupling
- *NotLockable*: Not lockable
- *Lockable*: Lockable coupling
- *FullyActive*: Fully active coupling (torque vectoring)

15.6 Power Supply

In CarMaker, Power Supply represents the vehicle's electric system and is part of the powertrain model. It includes all components of the vehicles low voltage and high voltage electric circuits and their batteries.

The Power Supply interface is designed for models with a power orientated point of view. That means, the consumer and generator loads are delivered as electric powers (and not split into voltage and electric current). The following sign convention is used: power consumption is positive; power generation is negative. The interface distinguishes up to three different electric circuits (LV, HV1, HV2). There may be a power transfer from HV1 to LV and/ or from HV1 to HV2, which is set by PTControl.

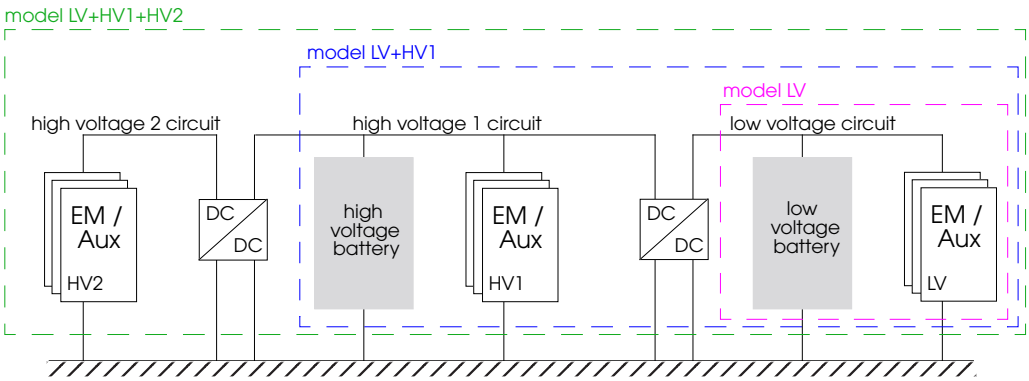


Figure 15.78: Supported PwrSupply variants

15.6.1 Power supply model

Overview

The primary task of the power supply subsystem is the accounting of the electric power flow to and between the electric circuits of the vehicle. It includes the battery models.

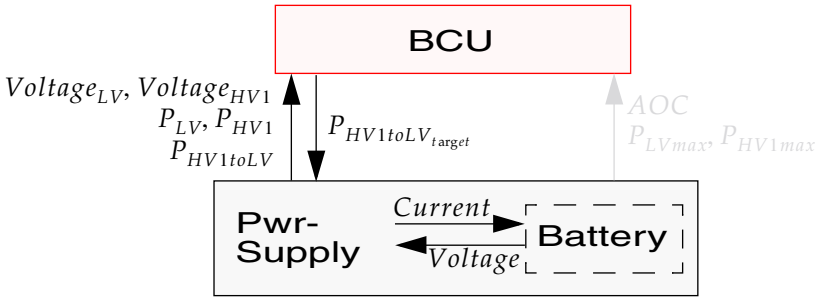


Figure 15.79: PwrSupply model

The states (e.g. voltage) of each electric circuit and battery are evaluated based on power consumption and generation of the electric motors and auxiliaries.

PowerTrain.PowerSupply.Kind = *KindStr VersionId*

Selection of power supply subsystem to use. The powertrain components library provides the following models:

ModelName	KindStr	Description
Low Voltage	LV	One low voltage electric circuit
Low Voltage + High Voltage 1	LV_HV1	One low voltage electric circuit and one high voltage electric circuit that are connected to each other with a DCDC converter
Low Voltage + High Voltage 1 + High Voltage 2	LV_HV1_HV2	One low voltage electric circuit and two high voltage electric circuits 1/2 that are connected to each other with a DCDC converter

Example `PowerTrain.PowerSupply.Kind = LV 1`

Each power supply model that is provided by the powertrain components library has its proper BCU model (see section 15.3.4 'Battery control unit (BCU)').

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tPTPowerSupplyCfgIF*:

Output Variable	Unit	Description
BattLV.Capacity BattHV.Capacity	Ah	Low / high battery capacity
BattLV.Voltage BattHV.Voltage	V	Low / high battery open circuit electric voltage
BattLV.SOC_min BattHV.SOC_min	%	Low / high battery minimum admitted state of charge
BattLV.SOC_max BattHV.SOC_max	%	Low / high battery maximum admitted state of charge

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tPTPowerSupplyIF*:

Input Variable	Unit	Description
Pwr_HV1toLV_trg	W	Target transferred electric power from high voltage 1 circuit to low voltage circuit
Pwr_LV	W	Total electric power (generators and consumers) on low voltage electric circuit
Pwr_HV1 Pwr_HV2	W	Total electric power (generators and consumers) on high voltage 1 / 2 electric circuit

Output Variable	Unit	Information
Pwr_HV1toLV	W	Instantaneous transferred electric power from high voltage 1 circuit to low voltage circuit

Output Variable	Unit	Information
Pwr_HV1toHV2	W	Instantaneous transferred electric power from high voltage 1 circuit to high voltage 2 circuit
Pwr_HV1toLV_max	W	Maximum possible transferred electric power from high voltage 1 electric circuit to low voltage electric circuit
Pwr_HV1toHV2_max	W	Maximum possible transferred electric power from high voltage 1 electric circuit to high voltage 2 electric circuit
Eta_HV1toLV	-	DC/DC efficiency from high voltage 1 electric circuit to low voltage electric circuit
Eta_HV1toHV2	-	DC/DC efficiency from high voltage 1 electric circuit to high voltage 2 electric circuit
Voltage_LV	V	Low voltage electric circuit instantaneous voltage
Voltage_HV1 Voltage_HV2	V	High voltage 1 / 2 electric circuit instantaneous voltage
BattLV.Current BattHV.Current	A	Low / high voltage battery electric current
BattLV.AOC BattHV.AOC	Ah	Low / high voltage battery instantaneous amount of charge
BattLV.Temp BattHV.Temp	K	Low / high voltage battery temperature
BattLV.Energy BattHV.Energy	kWh	Low / high voltage battery remaining energy capacity
BattLV.Pwr_max BattHV.Pwr_max	W	Low / high voltage battery maximum charge/discharge power

Power supply model "Low Voltage"

This power supply model contains only a low voltage circuit with a low voltage battery in it.

It evaluates the electric power taken from the low voltage circuit P_{LV} by summing the electric power of the auxiliary consumers/generators P_{AuxLV} and the consumer/generator power of each electric motor in this circuit $P_{MotorLV_i}$.

$$P_{LV} = P_{AuxLV} + \sum_i P_{MotorLV_i} \quad \text{(EQ 193)}$$

Furthermore, the electric current at the battery model I_{LV} is calculated under assumption that the electric circuit voltage U_{LV} is equal to its battery's voltage.

$$I_{LV} = \frac{P_{LV}}{U_{LV}} \quad \text{(EQ 194)}$$

In this model, electric power is only available when vehicle ignition is on.

Parameter Low Voltage

PowerTrain.PowerSupply.Pwr_LV_aux = *value*

Electric power of auxiliary consumers on the low voltage circuit. Default: 0W.

This parameter sets the initial electric power of auxiliary consumers.
The power consumption can be modified during the simulation using DVA on the quantity *PT.PwrSupply.LV.Pwr_aux* or directly the C-variable *PowerTrain.PowerSupply.Pwr_LV_aux*.

Power supply model "Low Voltage + High Voltage 1"

This power supply model contains one low voltage electric circuit and one high voltage electric circuit. These circuits are interconnected by a DC-DC converter and each one contains a battery model.

The electric power taken from each circuit P_{XV} depends on the electric power of the auxiliary consumers P_{AuxXV}, the consumer/generator power of each electric motor in this circuit $P_{MotorXV_i}$ and the power exchanged via the DC-DC converter $P_{HV1toLV}$. The transferred power between the circuits corresponds to the target transferred power between the circuits limited by the maximum DC-DC power. The model differentiates two cases:

$$P_{LV} = P_{AuxLV} + \sum_i P_{MotorLV_i} - P_{HV1toLV} \quad \text{(EQ 195)}$$

- The high voltage circuit charges the low voltage circuit.

$$P_{HV1toLV} \geq 0W \quad \text{(EQ 196)}$$

$$P_{HV1} = P_{AuxHV1} + \sum_i P_{MotorHV1_i} + \frac{P_{HV1toLV}}{\eta_{DCDC}} \quad \text{(EQ 197)}$$

- The low voltage circuit charges the high voltage circuit.

$$P_{HV1toLV} < 0W \quad \text{(EQ 198)}$$

$$P_{HV1} = P_{AuxHV1} + \sum_i P_{MotorHV1_i} + P_{HV1toLV} \cdot \eta_{DCDC} \quad \text{(EQ 199)}$$

Furthermore, the electric current at the battery model I_{XV} is calculated under assumption that the electric circuits voltage U_{XV} is equal to its battery's voltage.

$$I_{XV} = \frac{P_{XV}}{U_{XV}} \quad \text{(EQ 200)}$$

In this model, electric power is only available when vehicle ignition is on.

Parameter Low Voltage + High Voltage 1

PowerTrain.PowerSupply.Pwr_LV_aux = *value*
PowerTrain.PowerSupply.Pwr_HV1_aux = *value*

Electric power of auxiliary consumers on the low respectively high voltage 1 circuit.
Default: 0W.

This parameter sets the initial electric power of auxiliary consumers.
The power consumption can be modified during the simulation using DVA on the quantities *PT.PwrSupply.LV.Pwr_aux* / *PT.PwrSupply.HV1.Pwr_aux* or directly the C-variables *PowerTrain.PowerSupply.Pwr_LV_aux* / *PowerTrain.PowerSupply.Pwr_HV1_aux*.

PowerTrain.PowerSupply.DCDC_HV1_LV.Pwr_max = *value*

Maximum electric power that can pass the DC-DC converter. Default: 2.0kW.

PowerTrain.PowerSupply.DCDC_HV1_LV.EtaKind = *KindStr*

This model supports two subsystems to be selected.

ModelName	KindStr	Description
1D Look-Up Table	Map	DC-DC converter efficiency depends on the normalized electric power
Characteristic Value	Value	Constant DC-DC converter efficiency

Example `PowerTrain.PowerSupply.DCDC_HV1_LV.EtaKind = Map`

PowerTrain.PowerSupply.DCDC_HV1_LV.Eta = *value*

Efficiency of DC-DC converter for *PowerTrain.PowerSupply.DCDC_HV1_LV.EtaKind = Value*. Default: 1.0.

PowerTrain.PowerSupply.DCDC_HV1_LV.EtaMap: *Table*

Characteristic for the DC-DC converter efficiency mapping or *PowerTrain.PowerSupply.DCDC_HV1_LV.EtaKind = Map.* Vary normalized electric power $\left|\frac{P_{HV1toLV}}{P_{DCDCmax}}\right|$ from 0 to 1.

Syntax Infofile table mapping with 2 columns
<norm. electric power> <efficiency>

Example
```
PowerTrain.PowerSupply.DCDC_HV1_LV.EtaMap:
0.0     0.95
...
1.0     0.95
```

Power supply model "Low Voltage + High Voltage 1 + High Voltage 2"

This power supply model contains one low voltage electric circuit and two high voltage electric circuits. These circuits are interconnected by a DC-DC converter (see Figure 15.78) and the low voltage and high voltage 1 electric circuits contain a battery model.

The power exchange in the DC/DC converter between the high voltage 1 and low voltage electric circuit is the same as in the prevous model (see for details section 'Power supply model "Low Voltage + High Voltage 1"'). The power exchange in the DC/DC converter between the both high voltage circuits is calculated in similar manner:

$$P_{HV2} = P_{AuxHV2} + \sum_i P_{MotorHV2_i} - P_{HV1toHV2} \quad \text{(EQ 201)}$$

- The high voltage 1 circuit charges the high voltage 2 circuit.

$$P_{HV1toHV2} \geq 0W \quad \text{(EQ 202)}$$

$$P_{HV1} = P_{AuxHV1} + \sum_i P_{MotorHV1_i} + f(P_{HV1toLV}) + \frac{P_{HV1toHV2}}{\eta_{DCDC}} \quad \text{(EQ 203)}$$

- The high voltage 2 circuit charges the high voltage 1 circuit.

$$P_{HV1toHV2} < 0W \quad \textbf{(EQ 204)}$$

$$P_{HV1} = P_{AuxHV1} + \sum_i P_{MotorHV1_i} + f(P_{HV1toLV}) + P_{HV1toHV2} \cdot \eta_{DCDC} \quad \textbf{(EQ 205)}$$

Afterwards, the electric current at the low and high voltage battery model I_{XV} is calculated under assumption that the electric circuits voltage U_{XV} is equal to its battery's voltage.

$$I_{XV} = \frac{P_{XV}}{U_{XV}} \quad \textbf{(EQ 206)}$$

Parameter Low Voltage + High Voltage 1+ High Voltage 2

PowerTrain.PowerSupply.Pwr_LV_aux = *value*
PowerTrain.PowerSupply.Pwr_HV1_aux = *value*
PowerTrain.PowerSupply.Pwr_HV2_aux = *value*

Electric power of auxiliary consumers on the low respectively high voltage 1 / 2 circuits. Default: 0W.

This parameter sets the initial electric power of auxiliary consumers.
The power consumption can be modified during the simulation using DVA on the quantities *PT.PwrSupply.LV.Pwr_aux* / *PT.PwrSupply.HV1.Pwr_aux* / *PT.PwrSupply.HV2.Pwr_aux* or directly the C-variables *PowerTrain.PowerSupply.Pwr_LV_aux* / *PowerTrain.PowerSupply.Pwr_HV1_aux* / *PowerTrain.PowerSupply.Pwr_HV2_aux*.

In addition to the DC/DC converter between the low voltage and the high voltage 1 circuit (described in the previous model), a second one between the high voltage 1 and high voltage 2 circuit is required. It has the same parameters:

PowerTrain.PowerSupply.DCDC_HV1_HV2.Pwr_max = *value*
PowerTrain.PowerSupply.DCDC_HV1_HV2.EtaKind = *KindStr*
PowerTrain.PowerSupply.DCDC_HV1_HV2.Eta = *value*
PowerTrain.PowerSupply.DCDC_HV1_HV2.EtaMap: *Table*

For description details of these parameters see section 'Power supply model "Low Voltage + High Voltage 1"'.

PowerTrain.PowerSupply.DCDC_HV1_HV2.fac_Voltage = *value*

Specifies the voltage multiplication factor high voltage 2 / voltage 1. Default: 2.5.

Power supply model "User C-code / Simulink Plug-in / FMU"

Following parameters are required for the initialization of the output variables in the interface struct *tPTPowerSupplyCfgIF*:

PowerTrain.PowerSupply.BattLV.Active = *bool*
PowerTrain.PowerSupply.BattHV.Active = *bool*

Specifies if the low and high voltage batteries are implemented in the power supply model. Default: 1.

Depending on if the low and high voltage batteries are included, the following corresponding low or high voltage battery parameters are used:

PowerTrain.PowerSupply.BattLV.Capacity = *value*
PowerTrain.PowerSupply.BattHV.Capacity = *value*

Total capacity of the voltage battery. Default: 10Ah.

PowerTrain.PowerSupply.BattLV.Voltage_oc = *value*
PowerTrain.PowerSupply.BattHV.Voltage_oc = *value*

Battery open circuit (idle) voltage. Default: 12V.

PowerTrain.PowerSupply.BattLV.SOC_min = *value*
PowerTrain.PowerSupply.BattHV.SOC_min = *value*

Battery minimum admitted state of charge. Battery should not be discharged below this value. Default: 10%.

PowerTrain.PowerSupply.BattLV.SOC_max = *value*
PowerTrain.PowerSupply.BattHV.SOC_max = *value*

Battery maximum admitted state of charge. Battery should not be charged above this value. Default: 90%.

15.6.2 Battery

Overview

In CarMaker, the battery model is a subsystem of the power supply model. Its primary task is to reflect the electric energy storage in the powertrain. It provides a voltage to its electric circuit. The electric current from the battery is calculated externally by the power supply model.

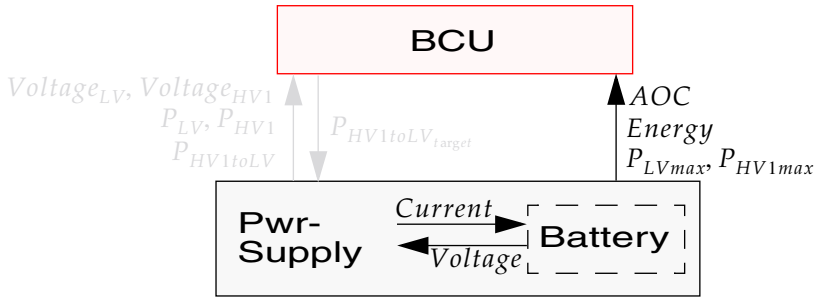


Figure 15.80: Battery model

Parameter and signal names of low voltage and high voltage battery are built the same way. This chapter uses the parameter and signal names of low voltage battery. The equivalent names for high voltage battery are obtained by replacing *BattLV* by *BattHV*.

PowerTrain.PowerSupply.BattLV.Kind = *KindStr VersionId*

Selection of low voltage battery subsystem to use. The powertrain components library provides the following models:

ModelName	KindStr	Description
Chen	Chen	Derived from Chen battery model

Example `PowerTrain.PowerSupply.BattLV.Kind = Chen 1`

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tPTBatteryCfgIF*:

Output Variable	Unit	Description
Capacity	Ah	Battery capacity
Voltage	V	Battery open circuit electric voltage
SOC_init	%	Battery initial state of charge
SOC_min	%	Battery minimum admitted state of charge
SOC_max	%	Battery maximum admitted state of charge

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tPTBatteryIF*:

Input Variable	Unit	Description
Current	A	Instantaneous battery electric current

Output Variable	Unit	Description
Voltage	V	Instantaneous battery electric voltage
AOC	Ah	Instantaneous battery amount of charge
Energy	kWh	Battery remaining energy capacity
Temp	K	Battery temperature
Pwr_max	W	Maximum charge/discharge power

Battery model "Chen"

This subsystem is derived from the battery model of Chen and Rincon-Mora that they describe in 'Accurate electrical battery model capable of predicting runtime and I-V-performance' (IEEE Transactions on Energy Conversion, vol. 21, 2006). It consists of the electric battery model depicted in Figure 15.81 and a characteristic curve that introduces the effect of the battery's state of charge on its voltage.

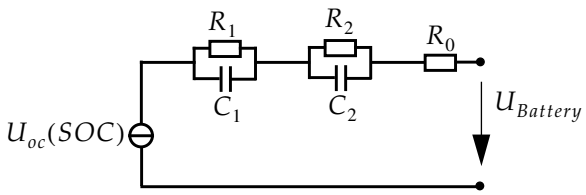


Figure 15.81: Battery model "Chen"

The equivalent circuit consists of a ideal voltage source with an open source voltage depending on battery's state of charge, two RC-circuits and a single resistance that are connected in series. Thus, the battery's terminal voltage is calculated the following way:

$$U_{Battery} = U_{oc}(SOC) - (I \cdot R_0) - U_1 - U_2 \quad \textbf{(EQ 207)}$$

with

$$U_1 = \int\left(\frac{I}{C_1} - \frac{U_1}{R_1 \cdot C_1}\right)dt \quad \textbf{(EQ 208)}$$

$$U_2 = \int\left(\frac{I}{C_2} - \frac{U_2}{R_2 \cdot C_2}\right)dt \quad \textbf{(EQ 209)}$$

and with the battery current I, calculated in the power supply model (see for details (EQ 194) and (EQ 200)).

Parameter Chen

PowerTrain.PowerSupply.BattLV.Capacity = *value*

Total capacity of the voltage battery. Default: 10Ah.

PowerTrain.PowerSupply.BattLV.R0 = *value*

Specifies the resistance R0 in the equivalent circuit. Default: 0.0012Ohm.

PowerTrain.PowerSupply.BattLV.R1 = *value*

Specifies the resistance R1 in the equivalent circuit. Default: 0.0004Ohm.

PowerTrain.PowerSupply.BattLV.R2 = *value*

Specifies the resistance R2 in the equivalent circuit. Default: 0.0011Ohm.

PowerTrain.PowerSupply.BattLV.C1 = *value*

Specifies the capacity C1 in the equivalent circuit. Default: 4074F.

PowerTrain.PowerSupply.BattLV.C2 = *value*

Specifies the capacity C2 in the equivalent circuit. Default: 82000F.

PowerTrain.PowerSupply.BattLV.Voltage_oc = *value*

Battery open circuit (idle) voltage. Default: 12V.

PowerTrain.PowerSupply.BattLV.SOC =*value*

Battery initial state of charge. Default: 70%.

PowerTrain.PowerSupply.BattLV.SOC_min = *value*

Battery minimum admitted state of charge. Battery should not be discharged below this value. Default: 10%.

PowerTrain.PowerSupply.BattLV.SOC_max = *value*

Battery maximum admitted state of charge. Battery should not be charged above this value. Default: 90%.

PowerTrain.PowerSupply.BattLV.VoltCorr_SOC.Amplify = *value*

Amplifies the output of the SOC correction factor characteristic by a given factor. Default: 1.

PowerTrain.PowerSupply.BattLV.VoltCorr_SOC: *Table*

One dimensional characteristic for the correction factor that defines the dependency of the battery's open circuit (idle) voltage from its state of charge.

Syntax Infofile table mapping with 2 columns
<state of charge> values 0...100 <voltage correction factor>

Example

```
PowerTrain.PowerSupply.BattLV.VoltCorr_SOC:
 0.0    0.0
  10    0.5
...
 100    1.1
```

PowerTrain.PowerSupply.BattLV.PwrMax.Kind = *kind*

Specifies the kind of maximum charge battery power. Possible kinds are:

- *Const*: By setting a constant maximum possible charge battery power (default).
- *ByCurrent*: By setting a maximum possible charge battery current.
- *DVA*: Setting the maximum possible charge battery power via DVA.

PowerTrain.PowerSupply.BattLV.PwrMax.Pwr_max = *value*

Specifies the constant maximum possible charge battery power. Default: 100kW.

PowerTrain.PowerSupply.BattLV.PwrMax.i_max = *value*

Specifies the constant maximum possible charge battery current. Default: 400A.

Battery model "User C-code / Simulink Plug-in / FMU"

The following parameters are required for the initialization of the output variables in the interface struct *tPTBatteryCfgIF*:

PowerTrain.PowerSupply.BattLV.Capacity = *value*

Total capacity of the voltage battery. Default: 10Ah.

PowerTrain.PowerSupply.BattLV.Voltage_oc = *value*

Battery open circuit (idle) voltage. Default: 12V.

PowerTrain.PowerSupply.BattLV.SOC =*value*

Battery initial state of charge. Default: 70%.

PowerTrain.PowerSupply.BattLV.SOC_min = *value*

Battery minimum admitted state of charge. Battery should not be discharged below this value. Default: 10%.

PowerTrain.PowerSupply.BattLV.SOC_max = *value*

Battery maximum admitted state of charge. Battery should not be charged above this value. Default: 90%.

15.7 Powertrain Models

15.7.1 Overview

The powertrain acts as a global system within the CarMaker VVE. It is composed of the powertrain modules that are described in section 'Powertrain'. These modules belong to five groups: General control strategy (PTControl), Control Units, Drive Sources, Driveline and Power Supply. Each powertrain model contains modules of each group, but the number and composition of modules may differ strongly.

That is why CarMaker integrates different powertrain model kinds that are listed below.

PowerTrain.Kind = *KindStr VersionId*

Selection of powertrain model kind to use. The powertrain components library provides the following models:

ModelName	KindStr	Drive Source	Description
Conventional	Generic	1	Conventional powertrain model with combustion engine and starter motor (see section 15.7.4 'Powertrain model "Generic"')
Parallel Hybrid	HEV_Parallel	1	Parallel hybrid powertrain with combustion engine, starter motor / one electric motor (only P2) (see section 15.7.5 'Powertrain model "Parallel Hybrid"')
AxleSplit Hybrid	HEV_AxleSplit	2-3	Hybrid powertrain with combustion engine on one axle and one/two electric motors on second axle (see section 15.7.6 'Powertrain model "Axle Split Hybrid"')
PowerSplit Hybrid	HEV_PowerSplit	1	PowerSplit hybrid powertrain with combustion engine, starter motor and one electric motor connected on a planet gear (see section 15.7.7 'Powertrain model "Power Split Hybrid"')
Serial Hybrid	HEV_Serial	1-4	Serial powertrain with combustion engine for charging battery and with up to 4 electric motors (see section 15.7.8 'Powertrain model "Serial Hybrid"')
Electrical	BEV	1-4	Electrical powertrain with up to 4 electric motors (see section 15.7.9 'Powertrain model "Electrical"')
OpenXWD	OpenXWD		User defined torque distribution to the wheels (see section 15.7.10 'Powertrain model "OpenXWD"')
AVL_CRUISE	AVL CRUISE		AVL CRUISE interface
GT_SUITE	GT-SUITE		GT-SUITE interface

Parameter Powertrain

All powertrain models (except AVL CRUISE and GT-SUITE) have one parameter in common.

PowerTrain.OSRate = *value*

Specifies the integration substeps in one simulation cycle of the whole powertrain model. The value is referred to 1ms cycle time. If another cycle time is specified, the integration substep is adapted internally to the specified cycle time. Default: 5.

15.7.2 Powertrain model interface

There are two interface structs defined for the information exchange between the powertrain model and the VVE, one for the initialization and one for the evaluation function.

The interface that is described in this chapter is the interface for the entire powertrain. OpenXWD Driveline only contains a Driveline model without integration of the wheel rotation. OpenXWD Powertrain contains a powertrain model without PTControl model and integration of the wheel rotation. Thus, their interfaces are slightly different (see section 15.7.10 'Powertrain model "OpenXWD"' and section 'Interface of "OpenXWD Powertrain"').

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tPowerTrainCfgIF*:

Input Variable	Unit	Description
nWheels		Number of vehicle wheels
WheelRadius	m	Mean wheel radius
Aero_Frcx	N	Aerodynamic drag force with velocity=1m/s and angle zero
Wheel_Iyy[*wh*]	kgm^2	Wheel *wh* inertia Iyy along spin axis

Output Variable	Unit	Description
StartEngineWithSST		1: Powertrain is activated with start-stop button 0: Powertrain is activated with key only
PTKind		Powertrain kind
GBKind		Gearbox kind (manual, automatic)
nFGears		Gearbox number of forward gears
iFGears[gear]		Gearbox ratios of forward gears
nBGears		Gearbox number of backward gears
iBGears[gear]		Gearbox ratios of backward gears
DL_iDiff_mean		Driveline mean differential value
Engine_rotv_max	rad/s	Engine maximum speed
Engine_rotv_idle	rad/s	Engine idle speed
Velocity_max	m/s	Vehicle maximum possible velocity for instruments

Interface for the evaluation function

Following table represents the input and output variables of the evaluation interface struct *tPowerTrainIF*:

Input Variable	Unit	Description
essential inputs		
Key	-	Vehicle key position
SST	bool	Activation start-stop button
Gas	-	Gas pedal position
Clutch	-	Clutch pedal position
Brake	-	Brake pedal position
GearNoTrg	-	Gearbox target gear (to overwrite TCU)
SelectorCtrl	-	Gearbox selector control
WheelIn[*wh*].Trq_Brake	Nm	Total conventional brake torque to the wheel *wh*
WheelIn[*wh*].Trq_T2W	Nm	Tire torque around wheel *wh* spin axis
additional inputs		
Velocity	m/s	Vehicle speed
UserSignal<*us*>	-	User defined signals *us*
WheelIn[*wh*].Trq_BrakeReg_trg	Nm	Target regenerative braking torque at wheel *wh* (absolute value) from brake control
WheelIn[*wh*].Trq_WhlBearing	Nm	Wheel *wh* bearing torque
DriveSrcIn[*ds*].Trq_trg	Nm	Target torque for drive source *ds* (e.g. intervention by brake control unit / ESP ECU)

Output Variable	Unit	Description
essential outputs		
Ignition	bool	Vehicle ignition
OperationState		Vehicle operation state
GearNo		Gearbox (behind engine) current gear
Engine_rotv	rad/s	Engine current rotation speed
WheelOut[*wh*].rotv	rad/s	Current wheel *wh* rotation speed
WheelOut[*wh*].Trq_B2W	Nm	Current reduced brake torque at wheel *wh*
WheelOut[*wh*].Trq_Supp2WC	Nm	Drive torque support on wheel carrier *wh*
additional outputs		
OperationError		Vehicle operation error or warning
Trq_Supp2Bdy1	Nm	Drive torque support on vehicle body Fr1A
Trq_Supp2Bdy1B	Nm	Drive torque support on vehicle body Fr1B
Trq_Supp2BdyEng	Nm	Drive torque support on engine body FrEng
DL_iDiff_mean		Driveline mean differential ratio (overrides the ratio from the initialization if non zero)
WheelOut[*wh*].rot	rad	Current wheel *wh* rotation angle
WheelOut[*wh*].Trq_Drive	Nm	Current drive torque at wheel *wh*
WheelOut[*wh*].Trq_BrakeReg	Nm	Estimated current regenerative brake torque at wheel *wh* (absolute value)

Output Variable	Unit	Description
WheelOut[*wh*].Trq_BrakeReg_max	Nm	Estimated maximum possible regenerative brake torque at wheel *wh* (absolute value)

Powertrain model "User C-code / Simulink Plug-in / FMU"

The following parameters are required for the initialization of the output variables in the interface struct *PowerTrainCfgIF*:

PowerTrain.PTKind = *PTKindStr*

Specifies the kind of the powertrain model. Following kinds are supported:

- *Generic*: Conventional powertrain with combustion engine as drive source
- *HEV_Parallel_P1*: Hybrid parallel powertrain without external clutch
- *HEV_Parallel_P2*: Hybrid parallel powertrain with external clutch
- *HEV_AxleSplit*: Hybrid axle split powertrain
- *HEV_PowerSplit*: Hybrid power split powertrain
- *HEV_Serial*: Hybrid serial powertrain
- *HEV_Any*: Any hybrid powertrain
- *BEV*: Full electrical powertrain

PowerTrain.Control.StartEngineWithSST = *bool*

Control element for switching operation states: 1 = switch operation states with start button, 0 = switch operation states with key only. Default: 1.

PowerTrain.Control.Velocity_max = *value*

Specifies the vehicle maximum possible velocity for instruments. Default: 260km/h.

PowerTrain.Engine.rotv_idle = *value*

Specifies the engine idle speed of the engine in the user powertrain model.
Default: 800rpm.

PowerTrain.Engine.rotv_max = *value*

Specifies the engine maximum speed of the engine in the user powertrain model.
Default: 8000rpm.

PowerTrain.GearBox.GBKind = *GBKindStr*

Specifies the kind of the gearbox used in the user powertrain model.
Following kinds are supported:

- *NoGearBox*: No gearbox
- *Manual*: Manual gearbox
- *AutoWithManual*: Automatic gearbox with optional manual gear target (Manumatic)
- *AutoNoManual*: Automatic gearbox without optional manual gear target or continuosly variable transmission (CVT)

PowerTrain.GearBox.iForwardGears = *GearRatioList*

State the transmission ratio for every single forward gear of the gearbox in the user powertrain model. Only required if manual gearbox or automatic gearbox with manual gear target are used.

Example `PowerTrain.GearBox.iForwardGears = 3.4 1.9 1.35 1.05 0.8`

PowerTrain.GearBox.iBackwardGears = *GearRatioList*

State the transmission ratio for every single backward gear of the gearbox in the user powertrain model. Only required if manual gearbox or automatic gearbox with manual gear target are used.

Example `PowerTrain.GearBox.iBackwardGears = -4.0`

PowerTrain.DL.iDiff_mean =*value*

Specifies the driveline mean differential ratio in the user powertrain model. Default: 3.0.

15.7.3 User Accessible Quantities for Powertrain

Please refer to section 24.12 'Powertrain'.

15.7.4 Powertrain model "Generic"

This powertrain model represents the structure of a conventional powertrain with one drive source. This drive source contains one combustion engine, one integrated starter motor, one (external or internal) clutch and one gearbox. The green point in Figure 15.82 marks the reference point for the total torque demand in IPGPowertrain PTControl models.

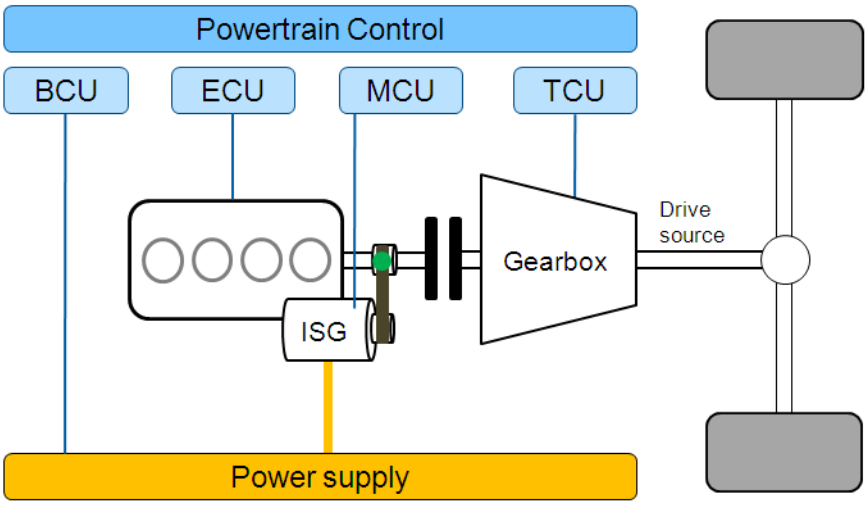


Figure 15.82: Powertrain model "Generic"

The powertrain model contains the modules listed in the table below:

Module Kind	Module	Information
PTControl	PTControl	To use with IPG models *Generic* or *Micro*
Control Units	ECU	
	MCU	
	TCU	
	BCU	
Drive Source	Engine	
	Starter Motor	
	Clutch	
	Gearbox	
Driveline	Driveline	
Power Supply	Power Supply	

PTControl model "Generic"

This PTControl model uses the powertrain model *Generic* as a conventional powertrain. That means the driving torque is exclusively delivered by the combustion engine and the starter motor only provides a torque until the combustion engine runs. Its output depends on the operation state:

- During operation state start, idle speed control is activated and the starter motor target load is set to 1.
- During operation state drive, the engine's target load is equal to the gas pedal position.

This PTControl model doesn't contain any battery or strategy management. PTControl model *Generic* has in general only the parameters of the operation states module (see section 15.2.4 'Operation states'). This is the default PTControl model.

PowerTrain.Control.GasInterpret.Active = *bool*

Optionally the PTControl model can use the "Interpretation of gas pedal position" to determine the target engine torque for ECU instead of load. Default: 0.
For more details please refer to section 15.2.5 'Interpretation of gas pedal position'.

PTControl model "Micro"

This PTControl model uses the powertrain model *Generic* as micro hybrid powertrain. It is built of the following PTControl modules:

- Operation states
- Interpretation of gas pedal position
- Strategy mode StartStop
- Strategy mode Boost
- Strategy mode EngineDrive
- Transient mode EngineStart, EngineStop
- Battery management

A description of these modules and their parameter (which also are the parameter of *Micro*) can be found in section 15.2 'PTControl'. Figure 15.83 depicts their composition. Here, the integrated starter motor adopts the tasks of the electric motor.

The reference point for the total torque demand is marked in Figure 15.82 respectively

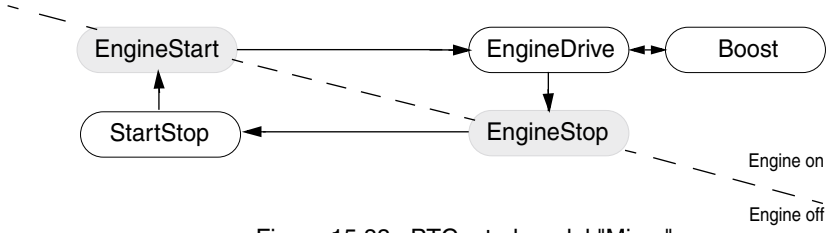


Figure 15.83: PTControl model "Micro"

This PTControl model supports the functionality either with a manual, or with an automatic transmission gearbox. For a proper operation between the PTControl and all powertrain modules, the model *Micro* prefers a special combination depending on gearbox type, which is listed below:

Module Kind	Module	Manual GearBox	Automatic with external Converter	Automat. Manual Transmission (AMT)
Control Units	ECU	Basic		
	MCU	Basic		
	TCU	-	Automatic	AMT
	BCU	Low Voltage		
Drive Source	Engine	Look-up Table (Mapping)		
	Starter Motor	Starter or Mapping on Low Voltage		
	Clutch	Friction	Converter	Closed
	Gearbox	Manual	Automatic	AMT
Driveline	Driveline	All models with one drive source (e.g. front, rear, all-wheel drive)		
Power Supply	Power Supply	Low Voltage		

15.7.5 Powertrain model "Parallel Hybrid"

This powertrain model contains one drive source that includes a combustion engine and one electric motor that is not a integrated starter motor. CarMaker distinguishes two configurations of this model type.

PowerTrain.Parallel_P2 = *bool*

Specifies if the parallel P2 powertrain configuration is used.
0: Parallel P1 (default), 1: Parallel P2.

Configuration "Parallel P1"

This powertrain model represents the structure of a parallel hybrid powertrain with one drive source. This drive source contains one combustion engine, one integrated starter motor and one gearbox with internal or external clutch. The green point in Figure 15.84 marks the reference point for the total torque demand in IG PTControl models.

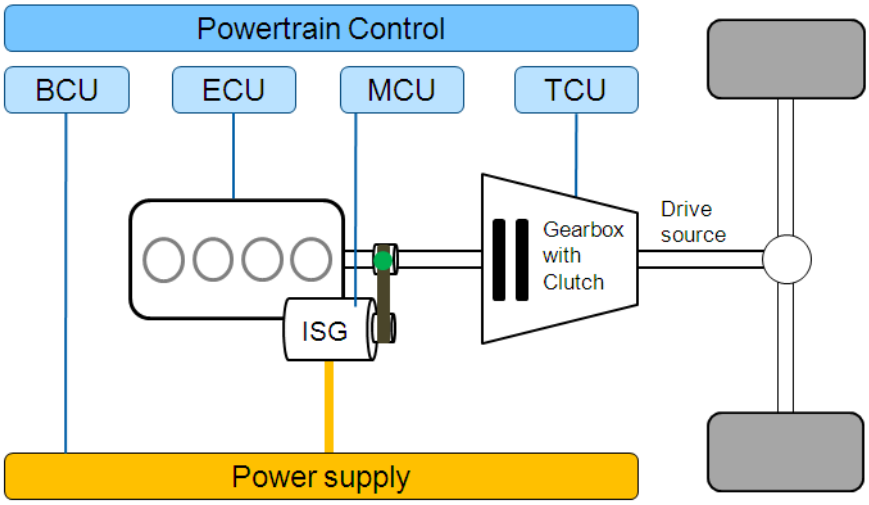


Figure 15.84: Powertrain model "Parallel P1"

The powertrain model contains the modules listed in the table below:

Module Kind	Module	Information
PTControl	PTControl	To use with IPGPowertrain model *Parallel P1*
Control Units	ECU	
	MCU	
	TCU	
	BCU	
Drive Source	Engine	
	Starter Motor	
	Clutch	
	Gearbox	
Driveline	Driveline	
Power Supply	Power Supply	

PTControl model "Parallel P1"

This PTControl model uses the powertrain model *Parallel* with configuration *P1* (one clutch) as mild hybrid powertrain. It is built of the following PTControl modules:

- Operation states
- Interpretation of gas pedal position
- Strategy mode StartStop
- Strategy mode RegBrake
- Strategy mode LoadShift
- Strategy mode Assist
- Strategy mode Boost
- Strategy mode EngineDrive
- Transient mode EngineStart, EngineStop
- Battery management

A description of these modules and their parameter (which also are the parameter of *Parallel P1*) can be found in section 15.2 'PTControl'. Figure 15.85 depicts their composition. Here, the integrated starter motor adopts the tasks of the electric motor. The reference point for the total torque demand is marked in Figure 15.84.

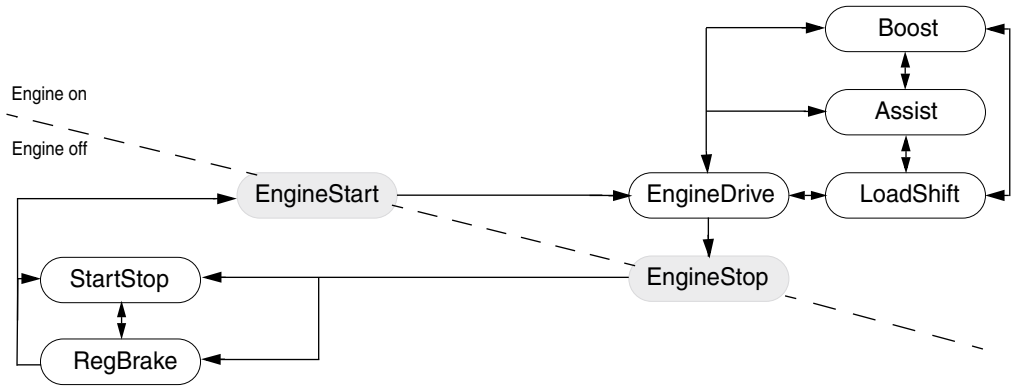


Figure 15.85: PTControl model "Parallel P1"

This PTControl model supports the functionality with an automatic transmission gearbox: automatic with internal converter or an automated manual transmission. For a proper operation between the PTControl and all powertrain modules, the model *Parallel P1* prefers a special combination depending on gearbox type, which is listed below:

Module Kind	Module	Automatic with internal Converter	Automated Manual Transmission (AMT)
Control Units	ECU	Basic	
	MCU	Basic	
	TCU	Automatic with Converter	Auto. Manual Transmission
	BCU	Low Voltage + High Voltage 1	
Drive Source	Engine	Look-up Table (Mapping) with active fuel consumption	
	Starter Motor	Mapping on High Voltage 1	
	Clutch	Closed	
	Gearbox	Automatic with Converter	Auto. Manual Transmission
Driveline	Driveline	All models with one drive source (e.g. front, rear, all-wheel drive)	
Power Supply	Power Supply	Low Voltage + High Voltage 1	

Configuration "Parallel P2"

This powertrain model represents the structure of a hybrid parallel powertrain with one drive source. This drive source contains one combustion engine, one integrated starter motor, one electric motor, one clutch and one gearbox with integrated clutch. The green point in Figure 15.86 marks the reference point for the total torque demand in IPGPowertrain PTControl models.

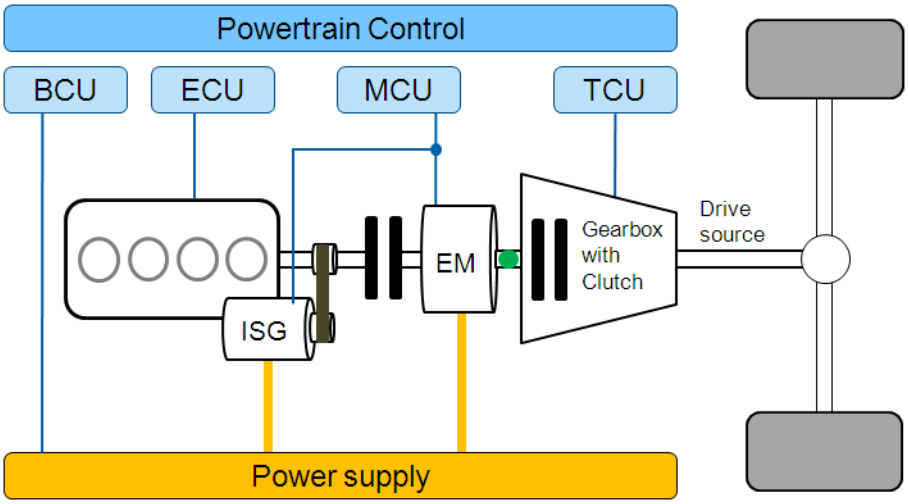


Figure 15.86: Powertrain model "Parallel P2"

The powertrain model contains the modules listed in the table below:

Module Kind	Module	Information
PTControl	PTControl	To use with IPGPowertrain model *Parallel P2*
Control Units	ECU	
	MCU	
	TCU	
	BCU	
Drive Source	Engine	
	Starter Motor	
	Motor	
	Clutch	
	Gearbox	
Driveline	Driveline	
Power Supply	Power Supply	

PTControl model "Parallel P2"

This PTControl model uses the powertrain model *Parallel* with configuration *P2* (two clutches) as full hybrid powertrain. It is built of the following PTControl modules:

- Operation states
- Interpretation of gas pedal position
- Strategy mode StartStop
- Strategy mode RegBrake
- Strategy mode RegDrag

- Strategy mode Coasting
- Strategy mode ElecDrive
- Strategy mode LoadShift
- Strategy mode Assist
- Strategy mode Boost
- Strategy mode EngineDrive
- Transient mode EngineStart, EngineStop
- Transient mode EngineSync
- Battery management

A description of these modules and their parameter (which also are the parameter of *Parallel P2*) can be found in section 15.2 'PTControl'. Figure 15.87 depicts their composition. The reference point for the total torque demand is marked in Figure 15.86.

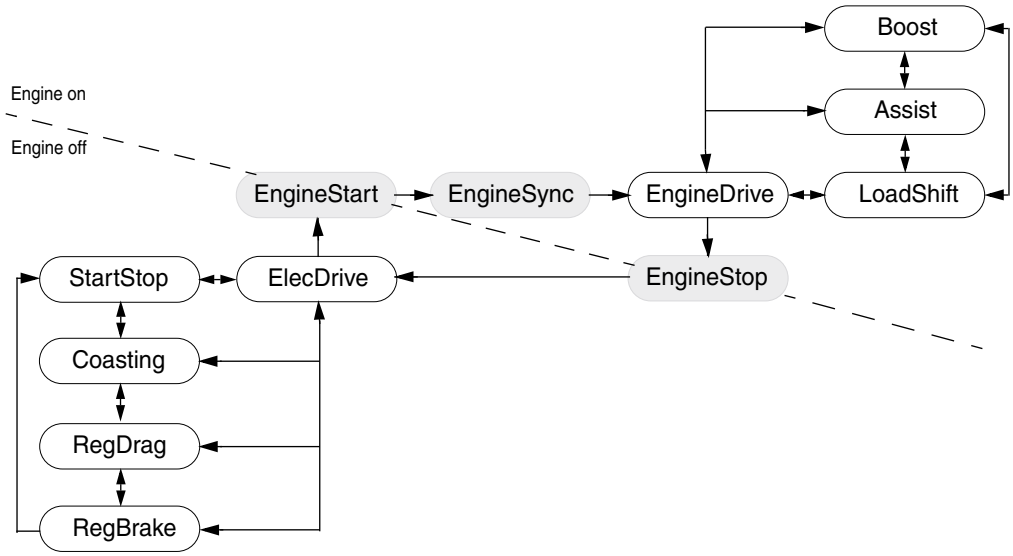


Figure 15.87: PTControl model "Parallel P2"

This PTControl model supports the functionality with an automatic transmission gearbox: automatic with internal converter or an automated manual transmission. For a proper operation between the PTControl and all powertrain modules, the model *Parallel P2* prefers a special combination depending on gearbox type, which is listed below:

Module Kind	Module	Automatic with internal Converter	Automated Manual Transmission (AMT)
Control Units	ECU	Basic	
	MCU	Basic	
	TCU	Automatic with Converter	Auto. Manual Transmission
	BCU	Low Voltage + High Voltage 1	
Drive Source	Engine	Look-up Table (Mapping) with active fuel consumption	
	Starter Motor	Starter or Mapping on Low Voltage	
	Motor	Mapping on High Voltage 1	
	Clutch	Friction	
	Gearbox	Automatic with Converter	Auto. Manual Transmission
Driveline	Driveline	All models with one drive source (e.g. front, rear, all-wheel drive)	
Power Supply	Power Supply	Low Voltage + High Voltage 1	

15.7.6 Powertrain model "Axle Split Hybrid"

This powertrain model represents the structure of a hybrid axle split powertrain (also known as "parallel through the road") with two until three possible drive sources. The first drive source contains the combustion engine, one integrated starter motor and one gearbox with integrated or separated clutch. On the second axle one or two additional drive sources (including an electric motor in combination with one gearbox with integrated clutch) can be defined. In this way it is possible to drive on the second axle via the differential or directly via the wheels or half shafts. The two possibilities are shown in Figure 15.88.

The green point in the figure marks the reference point for the total torque demand in IPG-Powertrain PTControl model.

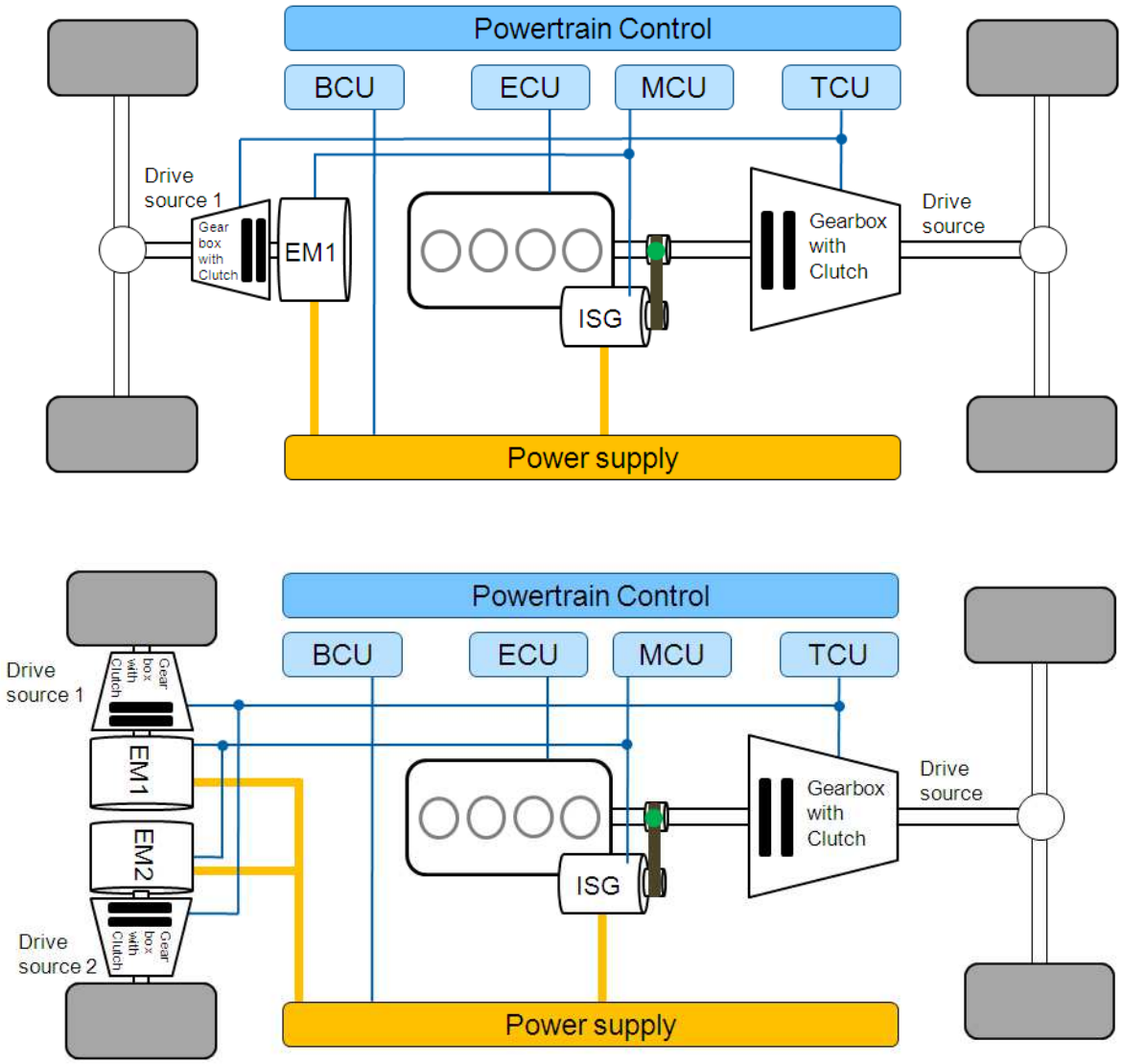


Figure 15.88: Powertrain model "Axle Split" variants with one and two electric motors

PowerTrain.nMotor = *value*

Specifies the number of electric motors. Default: 1.

The powertrain model contains the modules listed in the table below:

Module Kind	Module	Information
PTControl	PTControl	To use with IPGPowertrain model *AxleSplit*
Control Units	ECU	
	MCU	
	TCU	
	BCU	
Drive Source	Engine	
	Starter Motor	
	Clutch	
	Gearbox	
Drive Source 1/2	Motor 1/2	
	Gearbox Motor 1/2	
Driveline	Driveline	
Power Supply	Power Supply	

PTControl model "Axle Split"

This PTControl model uses the powertrain model *Axle Split* as full hybrid powertrain. It is built of the following PTControl modules:

- Operation states
- Interpretation of gas pedal position
- Strategy mode StartStop
- Strategy mode RegBrake
- Strategy mode RegDrag
- Strategy mode Coasting
- Strategy mode ElecDrive
- Strategy mode LoadShift
- Strategy mode Assist
- Strategy mode Boost
- Strategy mode EngineDrive
- Transient mode EngineStart, EngineStop
- Transient mode EngineSync
- Battery management

In this PTControl model the strategy mode Boost, Assist and LoadShift are executed by the integrated starter motor.

A description of these modules and their parameter (which also are the parameter of *Axle Split*) can be found in section 15.2 'PTControl'. Figure 15.89 depicts their composition. The reference point for the total torque demand is marked in Figure 15.88.

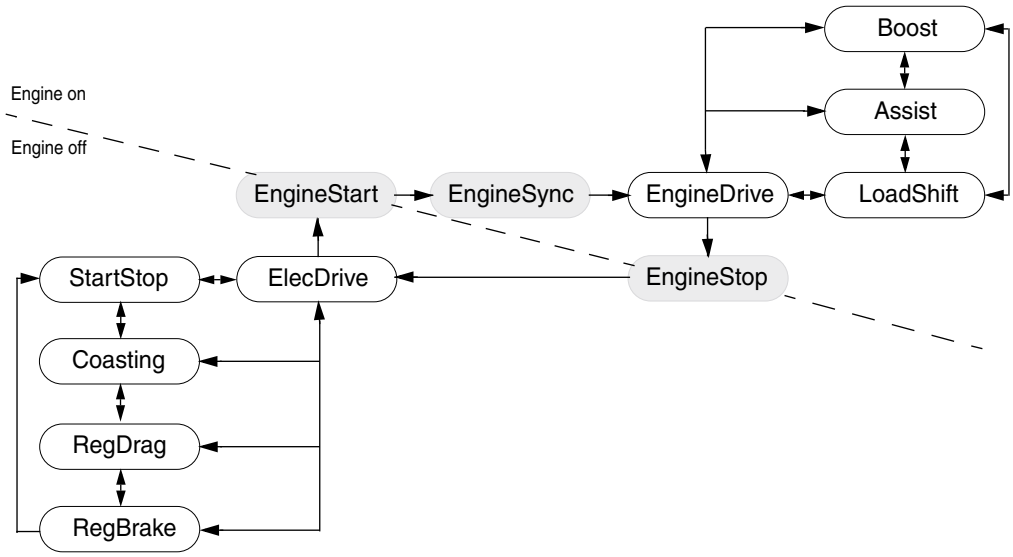


Figure 15.89: PTControl model "Axle Split"

This PTControl model supports the functionality with the automated manual transmission *AMT* or with automatic transmission with converter *Auto_Conv*. For a proper operation between the PTControl and all powertrain modules, the model *Axle Split* prefers a special combination depending on number of electric motors, which is listed below:

Configuration with Automated Manual Transmission

Following table illustrates the combination using the automatic manual transmission:

Module Kind	Module	Automated Manual Transmission (AMT)
Control Units	ECU	Basic
	MCU	Basic
	TCU	Automated Manual Transmission
	BCU	Low Voltage + High Voltage 1
Drive Source	Engine	Look-up Table (Mapping) with active fuel consumption
	Starter Motor	Mapping on High Voltage 1
	Clutch	Closed
	Gearbox	Automated Manual Transmission
Drive Source 1/2	Motor 1/2	Mapping on High Voltage 1
	Gearbox Motor 1/2	Automated Manual Transmission
Driveline	Driveline	nMotor=1: All models with 2 drive sources (front or rear drive with external torque to differential; universal drive with two drive sources) nMotor=2: All models with 3 drive sources (front or rear drive with external torque to wheels; universal drive with three drive sources)
Power Supply	Power Supply	Low Voltage + High Voltage 1

It is possible to use for the motor gearbox the model *Automated Manual Transmission* only with one gear ratio. Then the gearbox is handled as a fix ratio without shifting procedure. In this case the motor gearbox does not need to be controlled by the TCU.

Configuration with Automatic Gearbox with Converter

Following table illustrates the combination using the automatic gearbox with converter for the main gearbox (after engine) and using the automatic manual transmission for the motor gearbox.:

Module Kind	Module	Automatic with Converter (Auto_Conv)
Control Units	ECU	Basic
	MCU	Basic
	TCU	Automatic with Converter + AMT
	BCU	Low Voltage + High Voltage 1
Drive Source	Engine	Look-up Table (Mapping) with active fuel consumption
	Starter Motor	Mapping on High Voltage 1
	Clutch	Closed
	Gearbox	Automatic with Converter
Drive Source 1/2	Motor 1/2	Mapping on High Voltage 1
	Gearbox Motor 1/2	Automated Manual Transmission
Driveline	Driveline	nMotor=1: All models with 2 drive sources (front or rear drive with external torque to differential; universal drive with two drive sources) nMotor=2: All models with 3 drive sources (front or rear drive with external torque to wheels; universal drive with three drive sources)
Power Supply	Power Supply	Low Voltage + High Voltage 1

15.7.7 Powertrain model "Power Split Hybrid"

This powertrain model represents the structure of a hybrid power split powertrain with one drive source. In this special powertrain's architecture instead of a conventional clutch and gearbox a planetary gearbox is used. Figure 15.90 demonstrates how the powertrain modules are connected with the planetary gearbox: the combustion engine is connected to the planet carrier gear, the integrated starter motor is connected to the sun gear and the electric motor is connected to the ring gear.

The green point in the figure marks the reference point for the total torque demand in IPG-Powertrain PTControl model.

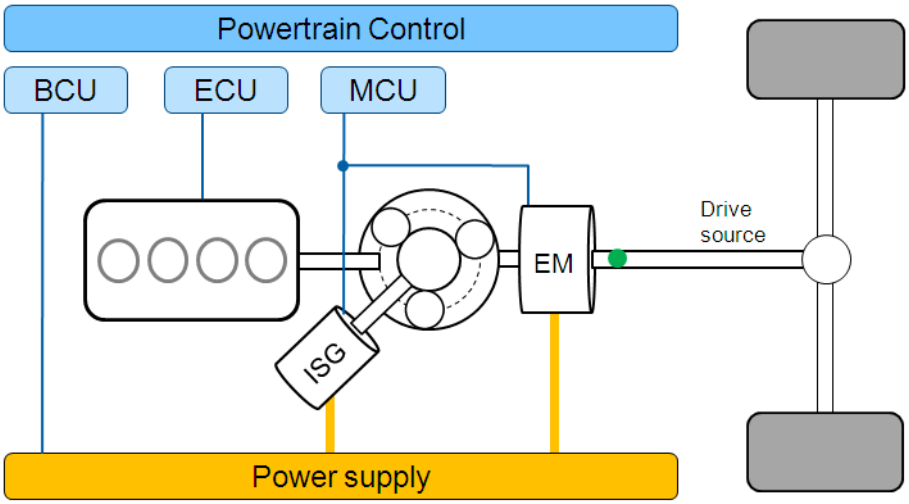


Figure 15.90: Powertrain model "Power Split"

The powertrain model contains the modules listed in the table below:

Module Kind	Module	Information
PTControl	PTControl	To use with IPGPowertrain model *Power Split*
Control Units	ECU	
	MCU	
	BCU	
Drive Source	Engine	
	Starter Motor	
	Motor	
	Planetary Gear	
Driveline	Driveline	
Power Supply	Power Supply	

The relationship in the planetary gearbox of the shaft speeds and moments is given by the *Willis formula* using the gear ratio i_G, which leads to the simplified equations below:

$$\omega_{Engine} = \frac{\omega_{ISG} + \omega_{Motor} \cdot i_G}{i_G + 1} \quad \text{(EQ 210)}$$

$$T_{DriveSrc} = T_{Motor} + \frac{T_{Engine} \cdot i_G}{i_G + 1} \quad \text{(EQ 211)}$$

PowerTrain.PlanetGear.I_ring = *value*

Specifies the inertia of the planetary gearbox ring gear. Default: 0.01kgm^2.

PowerTrain.PlanetGear.I_sun = *value*

Specifies the inertia of the planetary gearbox sun gear. Default: 0.002kgm^2.

PowerTrain.PlanetGear.I_cage = *value*

Specifies the inertia of the planetary gearbox cage (planet carrier) gear.
Default: 0.005kgm^2.

PowerTrain.PlanetGear.Ratio = *value*

Specifies the ratio of the planetary gearbox between the ring and sun gear. Default: 2.6.

PowerTrain.PlanetGear.Eta = *value*

Specifies the efficiency of the planetary gearbox between the ring and sun gear. Default: 1.

PTControl model "Power Split"

This PTControl model uses the powertrain model *Power Split* as full hybrid powertrain. It is built of the following PTControl modules:

- Operation states
- Interpretation of gas pedal position
- Strategy mode StartStop
- Strategy mode RegBrake
- Strategy mode RegDrag
- Strategy mode Coasting
- Strategy mode ElecDrive
- Strategy mode LoadShift
- Strategy mode Boost
- Strategy mode EngineDrive
- Transient mode EngineStart, EngineStop
- Battery management

A description of these modules and their parameter (which also are the parameter of *Power Split*) can be found in section 15.2 'PTControl'. Figure 15.91 depicts their composition. The reference point for the total torque demand is marked in Figure 15.90.

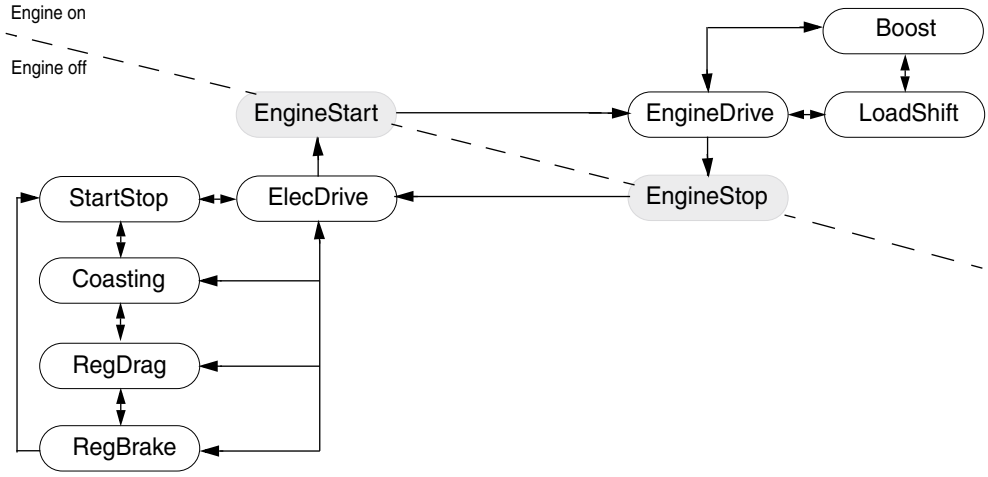


Figure 15.91: PTControl model "Power Split"

This PTControl model supports the functionality with the power supply using high voltage 2 circuit level. For a proper operation between the PTControl and all powertrain modules, the model *Power Split* prefers a special combination, which is listed below:

Module Kind	Module	Power supply with high voltage 2 circuit level
Control Units	ECU	Basic
	MCU	Basic
	BCU	Low Voltage + High Voltage 1 + High Voltage 2
Drive Source	Engine	Look-up Table (Mapping) with active fuel consumption
	Starter Motor	Mapping on High Voltage 2
	Motor	Mapping on High Voltage 2
Driveline	Driveline	All models with one drive source (e.g. front, rear, all-wheel drive)
Power Supply	Power Supply	Low Voltage + High Voltage 1 + High Voltage 2

15.7.8 Powertrain model "Serial Hybrid"

This powertrain model represents the structure of a hybrid serial powertrain (also known as "Range Extender") with one until four possible drive sources. In this powertrain's architecture the combustion engine is used to charge the battery and has no mechanical connection to the driven wheels. Figure 15.92 and Figure 15.93 demonstrates the different configurations with one up to four drive sources, which consist of an electric motor in combination with one gearbox with integrated clutch.

The green points in the figure marks the reference points for the total torque demand in IPG-Powertrain PTControl model.

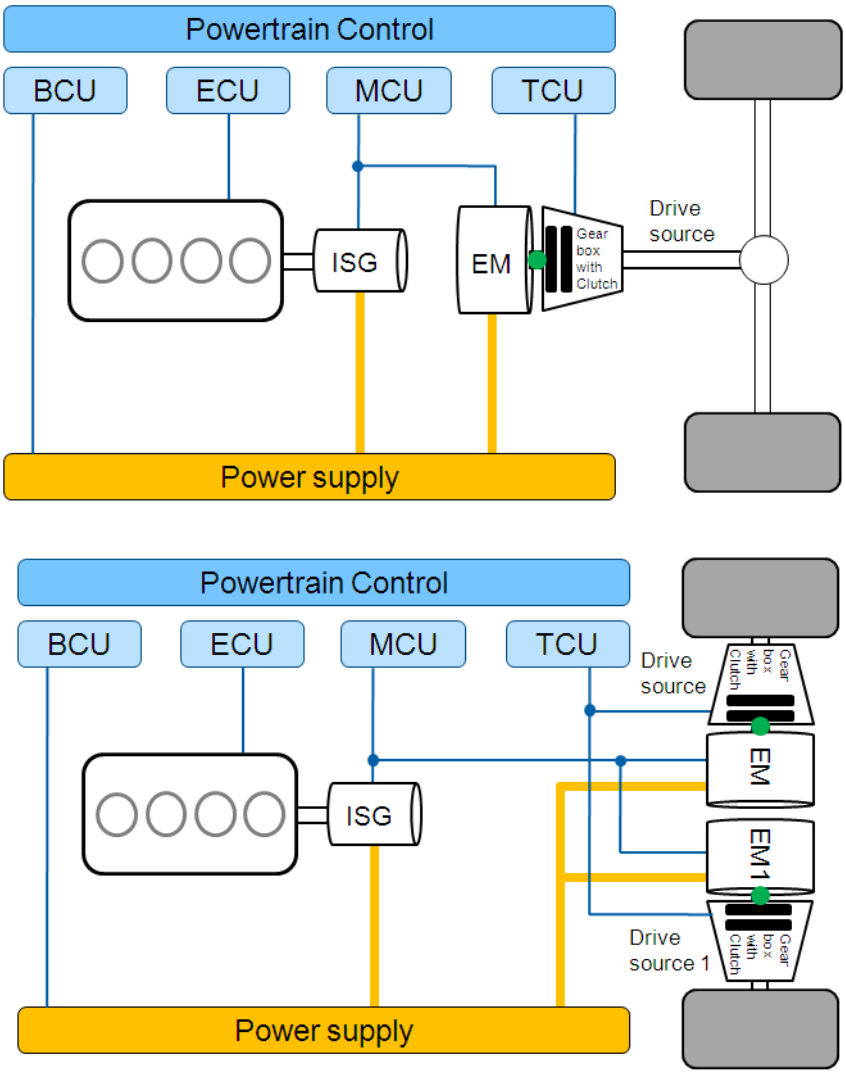


Figure 15.92: Powertrain model "Serial" variants with one and two electric motors

PowerTrain.nMotor = *value*

Specifies the number of electric motors. Default: 1.

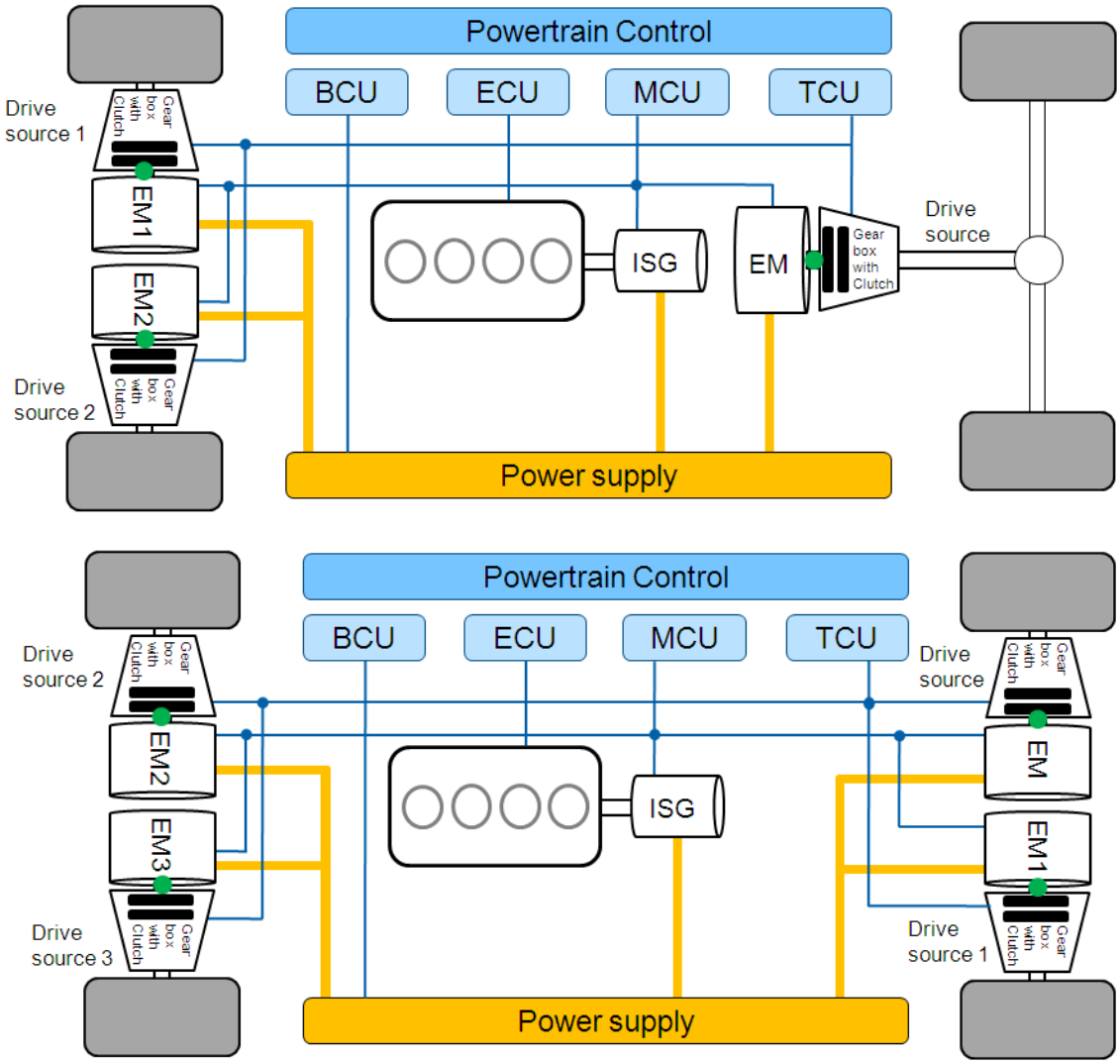


Figure 15.93: Powertrain model "Serial" variants with three and four electric motors

The powertrain model contains the modules listed in the table below:

Module Kind	Module	Information
PTControl	PTControl	To use with IPGPowertrain model *Serial*
Control Units	ECU	
	MCU	
	TCU	
	BCU	
Drive Source	Engine	
	Starter Motor	
	Motor	
	Gearbox Motor	
Drive Source 1-3	Motor 1-3	
	Gearbox Motor 1-3	
Driveline	Driveline	
Power Supply	Power Supply	

PTControl model "Serial"

This PTControl model uses the powertrain model *Serial* as full hybrid powertrain. It is built of the following PTControl modules:

- Operation states
- Interpretation of gas pedal position
- Strategy mode RegBrake
- Strategy mode RegDrag
- Strategy mode ElecDrive
- Strategy mode LoadShift
- Strategy mode Boost
- Strategy mode EngineDrive
- Transient mode EngineStart, EngineStop
- Battery management

Please note that the strategy modes EngineDrive, LoadShift and Boost are defined in a slightly different way for *Serial* powertrain because the combustion engine has no mechanic connection to the wheels. The vehicle is always driven by the electric motor(s). Nevertheless the energy flow of these strategy modes is similar to the other powertrain architecture's strategy modes of the same name:

- LoadShift (charge increasing):

  Engine is on and provides more electric energy than needed by the electric motor(s) so that the battery is charged.
- Boost (charge depleting):

  Engine is on and provides the maximum possible amount of electric energy; the additional demand of electric energy is covered by the battery.
- EngineDrive (charge sustaining):

  Engine is on and provides just the amount of electric energy needed by the electric motor(s).

A description of these modules and their parameter (which also are the parameter of *Serial*) can be found in section 15.2 'PTControl'. Figure 15.94 depicts their composition. The reference points for the total torque demand is marked in Figure 15.92 and Figure 15.93.

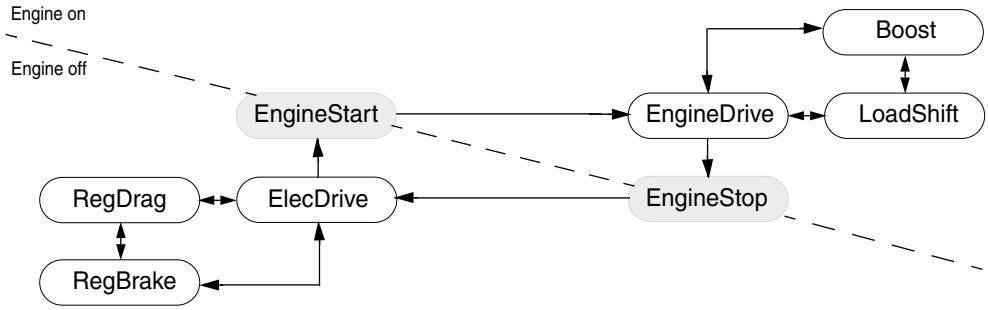


Figure 15.94: PTControl model "Serial"

PowerTrain.Control.TrqRatio_f = *value*

If using 2 or more than 2 electric motors to drive two axles (see Figure 15.93), the total torque distribution ratio to the front axle can be parametrized. Default: 0.5.

This PTControl model supports the functionality with the automated manual transmission. For a proper operation between the PTControl and all powertrain modules, the model *Serial* prefers a special combination depending on number of electric motors, which is listed below:

Module Kind	Module	Automated Manual Transmission (AMT)
Control Units	ECU	Basic
	MCU	Basic
	TCU	Automated Manual Transmission
	BCU	Low Voltage + High Voltage 1
Drive Source	Engine	Look-up Table (Mapping) with active fuel consumption
	Starter Motor	Mapping on High Voltage 1
	Motor	Mapping on High Voltage 1
	Gearbox Motor	Automated Manual Transmission
Drive Source 1-3	Motor 1-3	Mapping on High Voltage 1
	Gearbox Motor 1-3	Automated Manual Transmission
Driveline	Driveline	nMotor=1: All models with one drive source (front, rear, all-wheel drive; universal drive with one drive source) nMotor=2: universal drive with two drive sources on one axle or two axles nMotor=3: universal drive with three drive sources on two axles; front or rear drive with external torque to wheels nMotor=4: universal drive with four drive sources on two axles
Power Supply	Power Supply	Low Voltage + High Voltage 1

It is possible to use for the motor gearbox the model *Automated Manual Transmission* only with one gear ratio. Then the gearbox is handled as a fix ratio without shifting procedure. In this case the motor gearbox does not need to be controlled by the TCU.

15.7.9 Powertrain model "Electrical"

This powertrain model represents the structure of a electrical powertrain (also known as "BEV") with one until four possible drive sources. Figure 15.95 and Figure 15.96 demonstrates the different configurations with one up to four drive sources, which consist of an electric motor in combination with one gearbox with integrated clutch.

PowerTrain.nMotor = *value*

Specifies the number of electric motors. Default: 1.

The green points in the figure marks the reference points for the total torque demand in IPG-Powertrain PTControl model.

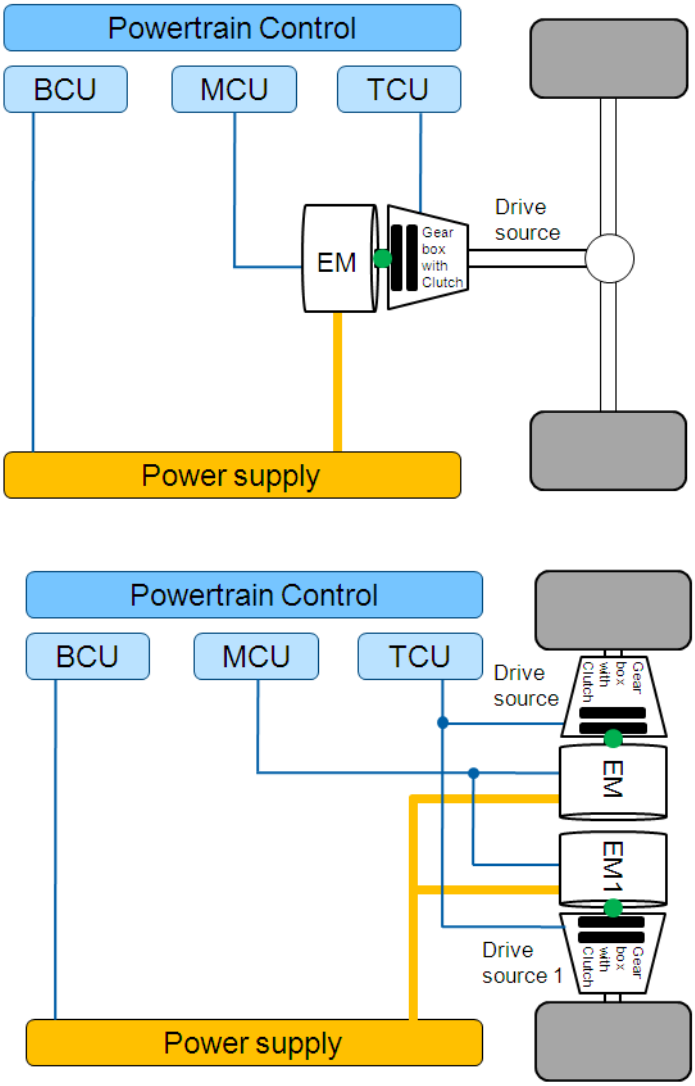


Figure 15.95: Powertrain model "Electrical" variants with one driven axle

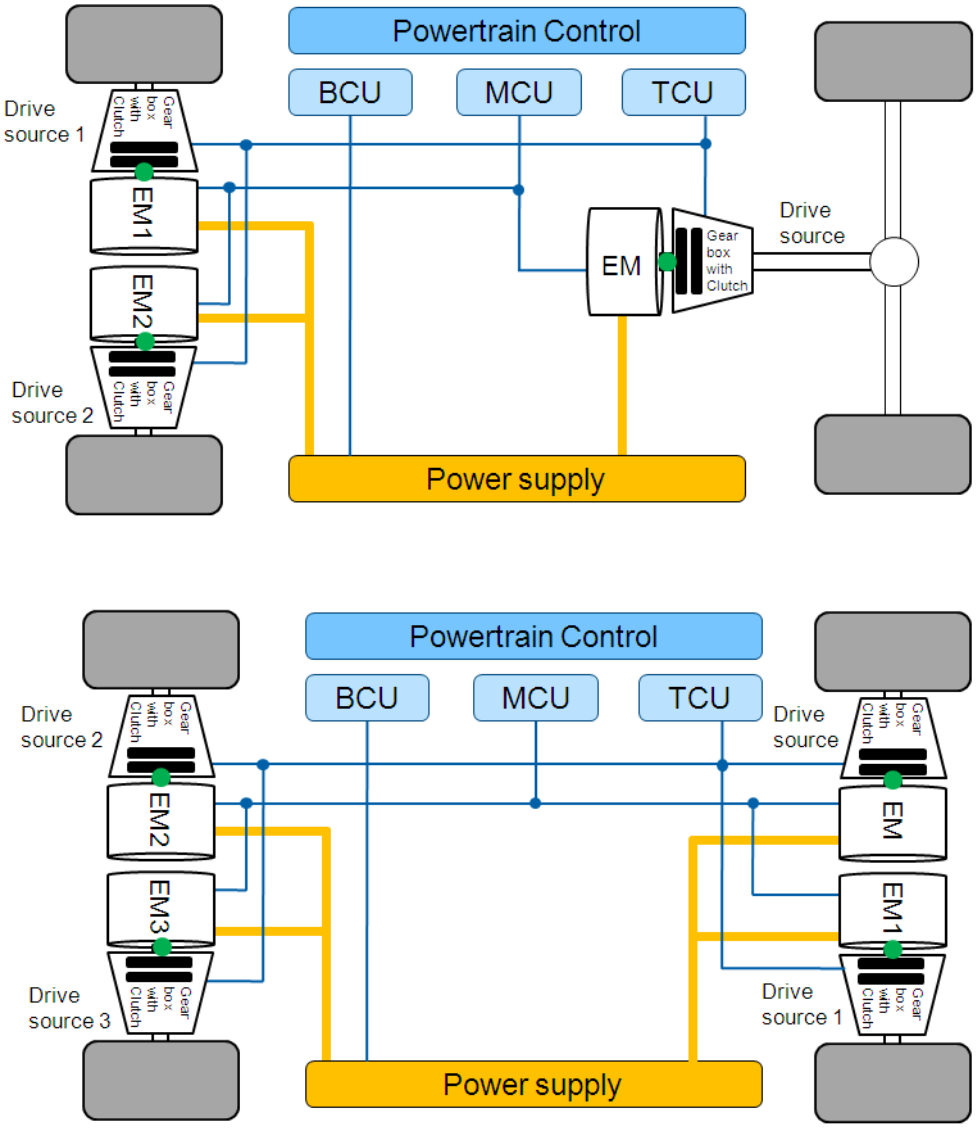


Figure 15.96: Powertrain model "Electrical" variants with two driven axles

The powertrain model contains the modules listed in the table below:

Module Kind	Module	Information
PTControl	PTControl	To use with IPGPowertrain model *Electrical*
Control Units	MCU	
	TCU	
	BCU	
Drive Source 1-4	Motor 1-4	
	Gearbox Motor 1-4	
Driveline	Driveline	
Power Supply	Power Supply	

PTControl model "Electrical"

This PTControl model uses the powertrain model *Electrical* as full electrical powertrain. It is built of the following PTControl modules:

- Operation states
- Interpretation of gas pedal position
- Strategy mode RegBrake
- Strategy mode RegDrag
- Strategy mode ElecDrive
- Battery management

A description of these modules and their parameter (which also are the parameter of *Electrical*) can be found in section 15.2 'PTControl'. Figure 15.97 depicts their composition. The reference points for the total torque demand is marked in Figure 15.95 and Figure 15.96.

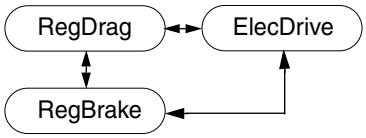


Figure 15.97: PTControl model "Electrical"

PowerTrain.Control.TrqRatio_f = *value*

If using 2 or more than 2 electric motors to drive two axles (see Figure 15.96), the total torque distribuition ratio to the front axle can be parametrized. Default: 0.5.

This PTControl model supports the functionality with the automated manual transmission. For a proper operation between PTControl and all powertrain modules, the model *Electrical* prefers a special combination depending on electric motors number, which is listed below:

Module Kind	Module	Automated Manual Transmission (AMT)
Control Units	MCU	Basic
	TCU	Automated Manual Transmission
	BCU	Low Voltage + High Voltage 1
Drive Source 1-4	Motor 1-4	Mapping on High Voltage 1
	Gearbox Motor 1-4	Automated Manual Transmission
Driveline	Driveline	nMotor=1: All models with one drive source (front, rear, all-wheel drive; universal drive with one drive source) nMotor=2: universal drive with two drive sources on one axle or two axles nMotor=3: universal drive with three drive sources on two axles; front or rear drive with external torque to wheels nMotor=4: universal drive with four drive sources on two axles
Power Supply	Power Supply	Low Voltage + High Voltage 1

It is possible to use for the motor gearbox the model *Automated Manual Transmission* only with one gear ratio. Then the gearbox is handled as a fix ratio without shifting procedure. In this case the motor gearbox does not need to be controlled by the TCU.

15.7.10 Powertrain model "OpenXWD"

Overview

The powertrain model *OpenXWD* gives the user the possibility to determine the distribution of drive torques to the wheels without having to manage the standstill behavior and the integration of the wheel rotation: this is done on the CarMaker's side. There are three different configurations to parametrize the drive source, which can be used for the calculation of the drive torques on the wheels within the user specific model.

PowerTrain.DL.Interface.Kind = *InterfaceStr*

Determines the configuration for the calculation of the drive source. Possible values are:

- *WithEngine*: The drive source is outside the OpenXWD driveline model on CarMaker's side, whose drive torque comes from the combustion engine (default).
- *WithMotor*: The drive source is outside the OpenXWD driveline model on CarMaker's side, whose drive torque comes from the electric motor.
- *StandAlone*: The drive source is within the OpenXWD powertrain model.

Configuration "With Engine"

With this configuration the user determines the drive torques on the wheels within the OpenXWD driveline model (for description of the interface please refer to section 'Interface of "OpenXWD Driveline"'). The drive source, whose drive torque can be used for the calculation of the drive torques on the wheels, is calculated by CarMaker. In this case the drive source consist of combustion engine in combination with gearbox with integrated or separated clutch (see Figure 15.98).

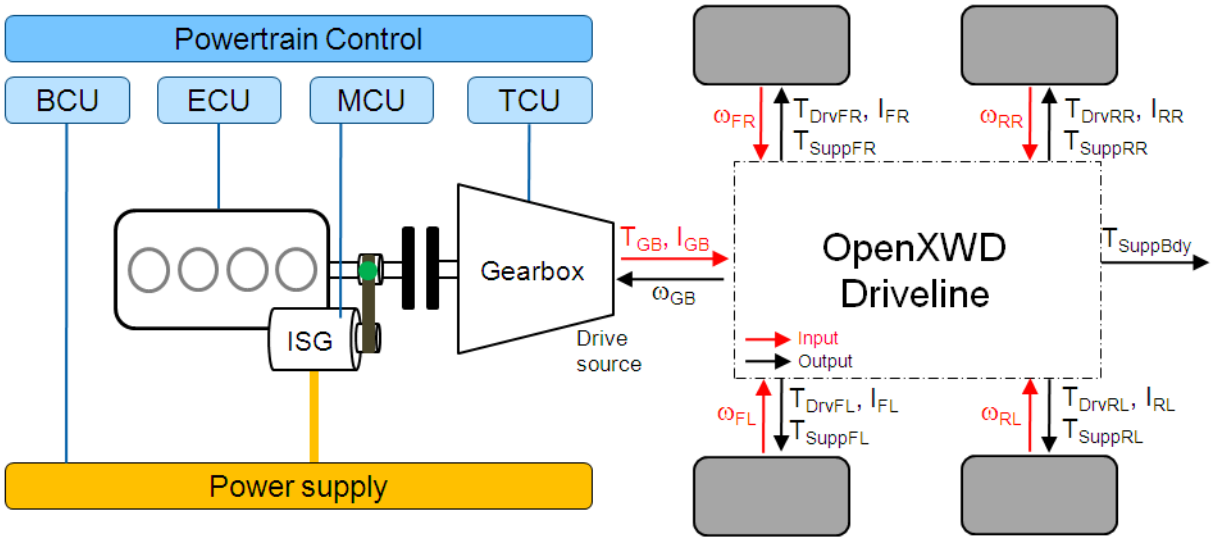


Figure 15.98: Powertrain model "OpenXWD Driveline" with combustion engine as drive source

With this configuration the powertrain model contains the modules listed in the table below:

Module Kind	Module	Information
PTControl	PTControl	Possible with *Generic* or *Micro*
Control Units	ECU	

Module Kind	Module	Information
	MCU	
	TCU	
	BCU	
Drive Source	Engine	
	Starter Motor	
	Clutch	
	Gearbox	
Driveline	Driveline	User specific *OpenXWD Driveline*
Power Supply	Power Supply	

Configuration "With Motor"

With this configuration the user determines the drive torques on the wheels within the OpenXWD driveline model (for description of the interface please refer to section 'Interface of "OpenXWD Driveline"'). The drive source, whose drive torque can be used for the calculation of the drive torques on the wheels, is calculated by CarMaker. In this case the drive source consist of an electric motor in combination with gearbox with integrated clutch (see Figure 15.99).

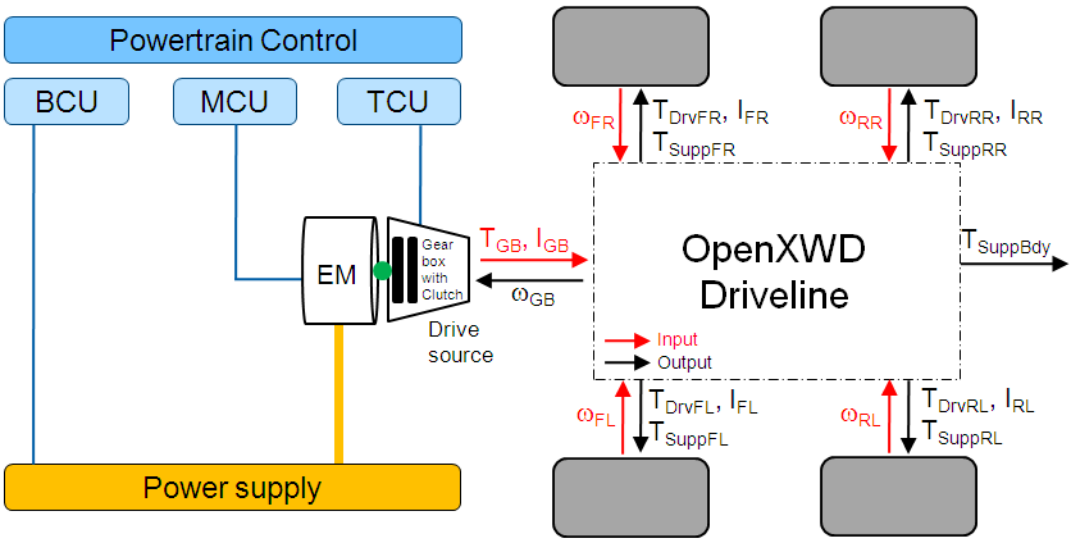


Figure 15.99: Powertrain model "OpenXWD Driveline" with electric motor as drive source

With this configuration the powertrain model contains the modules listed in the table below:

Module Kind	Module	Information
PTControl	PTControl	Possible with *Electrical*
Control Units	MCU	
	TCU	
	BCU	
Drive Source	Motor	
	Gearbox Motor	
Driveline	Driveline	User specific *OpenXWD Driveline*

Module Kind	Module	Information
Power Supply	Power Supply	

Configuration "Stand Alone"

With this configuration the user determines the drive torques on the wheels within the OpenXWD powertrain model (for description of the interface please refer to section 'Interface of "OpenXWD Powertrain"'). In this case the drive source is within the user specific model. The model can be controlled either by the corresponding control units or directly by PTControl (see Figure 15.100).

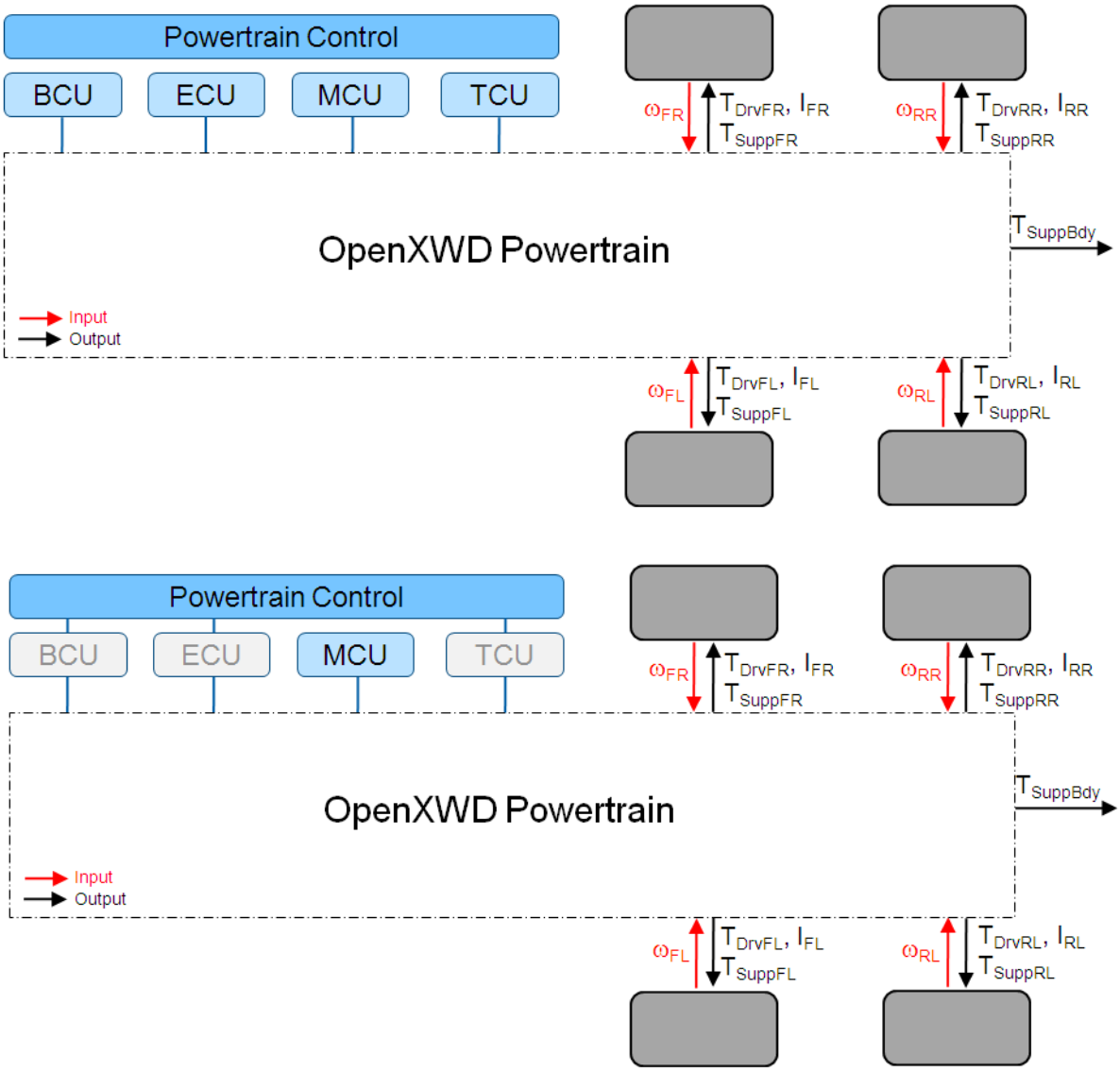


Figure 15.100: Powertrain model "OpenXWD Powertrain" with two possibilities for control units

With this configuration the powertrain model contains the modules listed in the table below:

Module Kind	Module	Information
PTControl	PTControl	depending on the *OpenXWD Powertrain* model
Control Units	ECU	optional
	MCU	optional

Module Kind	Module	Information
	TCU	optional
	BCU	optional
Driveline	Driveline	User specific *OpenXWD Powertrain*

Interface of "OpenXWD Driveline"

There are two interface structs defined for the information exchange between the powertrain model OpenXWD and the VVE, one for the initialization and one for the evaluation function.

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tPTDriveLineXWD_CfgIF*:

Input Variable	Information
nWheels	Number of vehicle wheels

Output Variable	Information
iDiff_mean	Driveline mean differential value

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tPTDriveLineXWD_IF*:

Input Variable	Unit	Information
DriveIn.Trq_in	Nm	Drive source input torque
DriveIn.Inert_in	kgm^2	Drive source input inertia
WheelIn[*wh*].rot	rad	Current wheel *wh* rotation angle
WheelIn[*wh*].rotv	rad/s	Current wheel *wh* rotation speed

Output Variable	Unit	Information
Trq_Supp2Bdy1	Nm	Drive torque support on vehicle body Fr1A
Trq_Supp2Bdy1B	Nm	Drive torque support on vehicle body Fr1B
Trq_Supp2BdyEng	Nm	Drive torque support on engine body FrEng
DriveOut.rot_in	rad	Drive source input shaft rotation angle
DriveOut.rotv_in	rad/s	Drive source input shaft rotation speed
iDiff_mean		Driveline mean differential ratio (overrides the ratio from the initialization if non zero)
WheelOut[*wh*].Inert_in	kgm^2	Wheel *wh* input inertia
WheelOut[*wh*].Trq_Drive	Nm	Current drive torque at wheel *wh*
WheelOut[*wh*].Trq_Supp2WC	Nm	Drive torque support on wheel *wh* carrier
PowerDelta.PlanetGear	W	Power delta in planetary gear
PowerDelta.Diffs	W	Total power delta in differentials
PowerDelta.Shafts	W	Power delta in driveline shafts
PowerDelta.Spring_DL	W	Stored power in driveline shafts spring element
PowerDelta.Inert_DL	W	Stored inertia power in whole driveline parts

Interface of "OpenXWD Powertrain"

There are two interface structs defined for the information exchange between the powertrain model OpenXWD and the VVE, one for the initialization and one for the evaluation function.

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tPowerTrainXWD_CfgIF*:

Input Variable	Unit	Information
nWheels		Number of vehicle wheels

Output Variable	Unit	Information
PTKind		Powertrain kind
ClKind		Clutch kind (friction, converter)
nMotor		Number of motors (without ISG)
nGearBoxM		Number of gearboxes (without gearbox for engine)
nPlanetGear		Number of planetary gears
DriveLine.iDiff_mean		Driveline mean differential value
DriveLine.DriveSourcePos[*ds*]		Specifies on which position the drive source number *ds* is applied
GearBox.GBKind GearBoxM<gb>.GBKind		Gearbox kind (manual, automatic)
GearBox.ClKind GearBoxM<*gb*>.ClKind		Gearbox clutch kind (friction, converter)
GearBox.nFGears GearBoxM<*gb*>.nFGears		Gearbox number of forward gears
GearBox.iFGear[gear] GearBoxM<*gb*>.iFGear[gear]		Gearbox ratios of forward gears
GearBox.nBGears GearBoxM<*gb*>.nBGears		Gearbox number of backward gears
GearBox.iBGear[gear] GearBoxM<*gb*>.iBGear[gear]		Gearbox ratios of backward gears
PlanetGear<*gb*>.Ratio		Planet gear *gb* ratio
BattLV.SOC_min BattHV.SOC_min	%	Low / high voltage battery minimum admitted state of charge
BattLV.SOC_max BattHV.SOC_max	%	High voltage battery maximum admitted state of charge
BattLV.Capacity BattHV.Capacity	Ah	Low / high battery capacity
BattLV.Voltage BattHV.Voltage	V	Low / high battery open circuit electric voltage
Engine.rotv_off	rad/s	Engine off speed
Engine.rotv_idle	rad/s	Engine idle speed
Engine.rotv_max	rad/s	Engine maximum speed
Engine.rotv_opt	rad/s	Engine optimum speed with optim. consumption

Output Variable	Unit	Information
Engine.Fuel_l2kWh	kWh/l	Factor for fuel conversion to kWh
Engine.TrqFull	Nm	1D-Lookup table for engine full load torque
Engine.TrqDrag	Nm	1D-Lookup table for engine drag torque
Engine.TrqOpt	Nm	1D-Lookup table for engine torque with optim. consumption as function of engine speed
ISG.Level Motor<*m*>.Level		PowerSupply voltage level (LV, HV1, HV2) of integrated starter generator / electric motor *m*
ISG.Ratio Motor<*m*>.Ratio		Ratio between motor shaft and driven shaft of integrated starter generator / electric motor *m*
ISG.rotv_Mot_max Motor<*m*>.rotv_Mot_max	rad/s	Maximum motor rotation speed of integrated starter generator / electric motor *m*
ISG.rotv_Gen_max Motor<*m*>.rotv_Gen_max	rad/s	Maximum generator rotation speed of integrated starter generator / electric motor *m*
ISG.TrqMot_max Motor<*m*>.TrqMot_max	Nm	1D-Lookup table for maximum motor torque of integrated starter generator / electric motor *m*
ISG.TrqGen_max Motor<*m*>.TrqGen_max	Nm	1D-Lookup table for maximum generator torque of integrated starter generator / electric motor *m*

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tPowerTrainXWD_IF*:

Input Variable	Unit	Information
Ignition	bool	Vehicle ignition
Gas	-	Gas pedal position
SelectorCtrl	-	Gearbox selector control
Velocity	m/s	Vehicle speed
EngineIn.set_ISC	bool	Idle speed controller activated
EngineIn.FuelCutOff	bool	Flag if fuel is cut-off
EngineIn.Load	-	Engine target load *
EngineIn.Trq_trg	Nm	Engine target torque
EngineIn.rotv_trg	rad/s	Engine target rotation speed
ISGIn.Load MotorIn<*m*>.Load	-	Integrated starter generator / motor *m* target load *
ISGIn.Trq_trg MotorIn<*m*>.Trq_trg	Nm	Integrated starter generator / motor *m* target torque
ISGIn.rotv_trg MotorIn<*m*>.rotv_trg	rad/s	Integrated starter generator / motor *m* target rotation speed
ClutchIn.Pos	-	Clutch target position
ClutchIn.rotv_out_trg	rad/s	Clutch output shaft target rotation speed
ClutchIn.Trq_out_trg	Nm	Clutch output shaft target torque
GearBoxIn.GearNoTrg GearBoxM_In<*gb*>.GearNoTrg	-	Gearbox target gear *
GearBoxIn.set_ParkBrake GearBoxM_In<*gb*>.set_ParkBrake	bool	Gearbox park brake set

Input Variable	Unit	Information
GearBoxIn.rotv_in_trg GearBoxM_In<*gb*>.rotv_in_trg	rad/s	Gearbox input shaft target rotation speed
GearBoxIn.Trq_out_trg GearBoxM_In<*gb*>.Trq_out_trg	Nm	Gearbox output shaft target torque
GearBoxIn.Clutch.Pos GearBoxM_In<*gb*>.Clutch.Pos	-	Gearbox clutch target position
GearBoxIn.Clutch.rotv_out_trg GearBoxM_In<*gb*>.Clutch.rotv_out_trg	rad/s	Gearbox clutch output shaft target rotation speed
GearBoxIn.Clutch.Trq_out_trg GearBoxM_In<*gb*>.Clutch.Trq_out_trg	Nm	Gearbox clutch output shaft target torque
PwrSupplyIn.Pwr_HV1toLV_trg	W	Target transferred electric power from high voltage1 circuit to low voltage circuit
WheelIn[*wh*].rot	rad	Wheel *wh* current rotation angle
WheelIn[*wh*].rotv	rad/s	Wheel *wh* current rotation speed

The input signals marked with * are thought to overwrite the appropriated values calculated by the corresponding control units (ECU, MCU, TCU).

Output Variable	Unit	Information
ECU_Status	-	Engine control unit status
EngineOut.Engine_on	bool	Flag is engine is on
EngineOut.rotv	rad/s	Current engine rotation speed
EngineOut.Trq	Nm	Current engine torque
EngineOut.TrqDrag	Nm	Engine drag torque at current speed
EngineOut.TrqFull	Nm	Engine full load torque at current speed
EngineOut.TrqOpt	Nm	Engine torque with optimal consumption at current speed
EngineOut.FuelFlow	l/s	Current fuel flow
MCU_Status	-	Motor control unit status
ISGOut.rotv MotorOut<*m*>.rotv	rad/s	Current integrated starter generator / motor *m* rotation speed
ISGOut.Trq MotorOut<*m*>.Trq	Nm	Current integrated starter generator / motor *m* torque
ISGOut.TrqMot_max MotorOut<*m*>.TrqMot_max	Nm	Integrated starter generator / motor *m* maximum motor torque on driven shaft
ISGOut.TrqGen_max MotorOut<*m*>.TrqGen_max	Nm	Integrated starter generator / motor *m* max. generator torque on driven shaft
ISGOut.PwrElec MotorOut<*m*>.PwrElec	W	Current integrated starter generator / motor *m* electric power
ISGOut.i_M2W<*pos*> MotorOut<*m*>.i_M2W<pos>	-	Theoretical ratio from integrated starter generator / motor *m* to wheel without considering friction clutch
TCU_Status	-	Transmission control unit status
ClutchOut.Pos	-	Clutch position
ClutchOut.rotv_in	rad/s	Clutch input shaft rotation speed
ClutchOut.rotv_out	rad/s	Clutch output shaft rotation speed

Output Variable	Unit	Information
ClutchOut.Trq_in	Nm	Clutch input shaft torque
ClutchOut.Trq_out	Nm	Clutch output shaft torque
ClutchOut.i_TrqIn2Out	-	Ratio between clutch input shaft torque and output shaft torque
GearBoxOut.GearNo GearBoxM_Out<*gb*>.GearNo	-	Gearbox current gear
GearBoxOut.Trq_DriveSrc_trg GearBoxM_Out<gb>.Trq_DriveSrc_trg	Nm	Optional drive source target torque from TCU to PTControl (e.g. while shifting)
GearBoxOut.i GearBoxM_Out<*gb*>.i	-	Gearbox current ratio
GearBoxOut.rotv_in GearBoxM_Out<*gb*>.rotv_in	rad/s	Gearbox input shaft rotation speed
GearBoxOut.rotv_out GearBoxM_Out<*gb*>.rotv_out	rad/s	Gearbox output shaft rotation speed
GearBoxOut.Trq_in GearBoxM_Out<*gb*>.Trq_in	Nm	Gearbox input shaft torque
GearBoxOut.Trq_out GearBoxM_Out<*gb*>.Trq_out	Nm	Gearbox output shaft torque
GearBoxOut.Clutch.Pos GearBoxM_Out<*gb*>.Clutch.Pos	-	Gearbox clutch position
GearBoxOut.Clutch.rotv_in GearBoxM_Out<*gb*>.Clutch.rotv_in	rad/s	Gearbox clutch input shaft rotation speed
GearBoxOut.Clutch.rotv_out GearBoxM_Out<*gb*>.Clutch.rotv_out	rad/s	Gearbox clutch output shaft rotation speed
GearBoxOut.Clutch.Trq_in GearBoxM_Out<*gb*>.Clutch.Trq_in	Nm	Gearbox clutch input shaft torque
GearBoxOut.Clutch.Trq_out GearBoxM_Out<*gb*>.Clutch.Trq_out	Nm	Gearbox clutch output shaft torque
GearBoxOut.Clutch.i_TrqIn2Out GearBoxM_Out<*gb*>.Clutch.i_TrqIn2Out	-	Ratio between gearbox clutch input shaft torque and output shaft torque
BCU_Status	-	Battery control unit status
BattLVOut.SOC BattHVOutV.SOC	%	Low / high voltage battery instantaneous state of charge
BattLVOut.SOH BattHVOut.SOH	%	Low / high voltage battery state of health
BattLVOut.Current BattHVOut.Current	A	Low / high voltage battery electric current
BattLVOut.AOC BattHVOut.AOC	Ah	Low / high voltage battery instantaneous amount of charge
BattLVOut.Temp BattHVOut.Temp	K	Low / high voltage battery instantaneous temperature
BattLVOut.Energy BattHVOut.Energy	kWh	Low / high voltage battery remaining energy capacity
BattLVOut.Pwr_max BattHVOut.Pwr_max	W	Low / high voltage battery maximum charge / recharge power
PwrSupplyOut.Pwr_LV	W	Total electric power (consumers and generators) on low voltage electric circuit

Output Variable	Unit	Information
PwrSupplyOut.Pwr_HV1 PwrSupplyOut.Pwr_HV2	W	Total electric power (consumers and generators) on high voltage 1 / 2 electric circuit
PwrSupplyOut.Voltage_LV	V	Low voltage electric circuit instantaneous voltage
PwrSupplyOut.Voltage_HV1 PwrSupplyOut.Voltage_HV2	V	High voltage 1 / 2 electric circuit instantaneous voltage
PwrSupplyOut.Pwr_HV1toLV	W	Instantaneous transferred electric power from high voltage 1 circuit to low voltage circuit
PwrSupplyOut.Pwr_HV1toHV2	W	Instantaneous transferred electric power from high voltage 1 electric circuit to high voltage 2 electric circuit
PwrSupplyOut.Pwr_HV1toLV_max	W	Maximum possible transferred electric power from high voltage 1 electric circuit to low voltage electric circuit
PwrSupplyOut.Pwr_HV1toHV2_max	W	Maximum possible transferred electric power from high voltage 1 electric circuit to high voltage 2 electric circuit
PwrSupplyOut.Eta_HV1toLV	-	DC/DC efficiency from high voltage 1 electric circuit to low voltage electric circuit
PwrSupplyOut.Eta_HV1toHV2	-	DC/DC efficiency from high voltage 1 electric circuit to high voltage 2 electric circuit
Trq_Supp2Bdy1	Nm	Drive torque support on vehicle body Fr1A
Trq_Supp2Bdy1B	Nm	Drive torque support on vehicle body Fr1B
Trq_Supp2BdyEng	Nm	Drive torque support on vehicle body FrEng
DL_iDiff_mean		Driveline mean differential ratio (overrides the ratio from the initialization if non zero)
WheelOut[*wh*].Inert_in	kgm^2	Wheel *wh* input inertia
WheelOut[*wh*].Trq_Drive	Nm	Current driving torque at wheel *wh*
WheelOut[*wh*].Trq_Supp2WC	Nm	Drive torque support on wheel *wh* carrier
PowerDelta.PlanetGear	W	Power delta in planetary gear
PowerDelta.Diffs	W	Total power delta in differentials
PowerDelta.Shafts	W	Power delta in driveline shafts
PowerDelta.Spring_DL	W	Stored power in driveline shafts spring element
PowerDelta.Inert_DL	W	Stored inertia power in all driveline parts
PowerDelta.PowerSupply	W	Power delta in power supply

Parameters

The user has the possibility to implement his OpenXWD driveline/powertrain model either as C-model (e.g. Simulink RTW-C source code) using the model manager, which calculates and sets the required signals in the interface struct, or to set the signals with Direct Variable Access (DVA) by manipulating the corresponding User Accessible Quantities (UAQs).

User Accessible Quantities are updated outside of the powertrain and only once per cycle (generally 1ms). On the other hand a driveline model registered using the model manager can be updated more times per cycle because of the ‘integration substeps’ feature of the powertrain. On this account, it’s better to use a proper driveline model, to improve numerical stability.

Parameters for Configuration "With Engine" and "WithMotor"

PowerTrain.DLXWD.Kind = *KindStr VersionId*

Optional. Selects the OpenXWD user driveline model for calculating the propulsion torques. Default: none, the propulsion torques are manipulated with UAQ.

DrivelineXWD model "User C-code / Simulink Plug-in / FMU / DVA"

The following parameters are required for the initialization of the output variables in the interface struct *tPTDriveLineXWD_CfgIF* if using a user model or using DVA:

PowerTrain.DL.iDiff_mean =*value*

Specifies the driveline mean differential ratio in the user model. Default: 3.0.

Parameters for Configuration "Stand Alone"

PowerTrain.PTXWD.Kind = *KindStr VersionId*

Optional. Selects the OpenXWD user powertrain model for calculating the propulsion torques. Default: none, the propulsion torques are manipulated with UAQ.

PowertrainXWD model "User C-code / Simulink Plug-in / FMU / DVA"

The following parameters are required for the initialization of the output variables in the interface struct *tPowerTrainXWD_CfgIF* if using a C-model or using DVA:

PowerTrain.PTKind = *PTKindStr*

Specifies the kind of the OpenXWD powertrain model (no kind as default).
Following kinds are supported:

- *Generic*: Conventional powertrain with combustion engine as drive source
- *HEV_Parallel_P1*: Hybrid parallel powertrain without external clutch
- *HEV_Parallel_P2*: Hybrid parallel powertrain with external clutch
- *HEV_AxleSplit*: Hybrid axle split powertrain
- *HEV_PowerSplit*: Hybrid power split powertrain
- *HEV_Serial*: Hybrid serial powertrain
- *HEV_Any*: Any hybrid powertrain
- *BEV*: Full electrical powertrain

PowerTrain.Engine.Fuel_l2kWh = *value*

Specifies the conversion factor for fuel from liter to kWh. Default: 8.48 kWh/l.

PowerTrain.Engine.rotv_idle = *value*

Specifies the engine idle speed. Default: 800rpm.

PowerTrain.Engine.rotv_off = *value*

Specifies the engine off speed. Default: 500rpm.

PowerTrain.Engine.rotv_max = *value*

Specifies the engine maximum speed. Default: 8000rpm.

PowerTrain.Engine.rotv_opt = *value*

Specifies the engine speed with optimal fuel consumption. Default: 2300rpm.

PowerTrain.Engine.TrqFull: *Table*

Optional. One dimensional characteristic mapping for the maximum engine full torque.

Syntax Infofile table mapping with 2 columns
<engine speed> [rpm] <engine full torque> [Nm]

Example
```
PowerTrain.Engine.TrqFull:
500.0        10.0
1000.0       140.0
...
```

PowerTrain.Engine.TrqDrag: *Table*

Optional. One dimensional characteristic mapping for the maximum engine drag torque.

Syntax Infofile table mapping with 2 columns
<engine speed> [rpm] <engine drag torque> [Nm]

Example
```
PowerTrain.Engine.TrqDrag:
500.0        -10.0
1000.0      -10.0
...
```

PowerTrain.Engine.TrqOpt: *Table*

Optional. One dimensional characteristic mapping for engine torque with optimal consumption (minimal specific fuel consumption as function of engine speed).

Syntax Infofile table mapping with 2 columns
<engine speed> [rpm] <engine optimal torque> [Nm]

Example
```
PowerTrain.Engine.TrqOpt:
500.0      80.0
1000.0    110.0
...
```

PowerTrain.Clutch.ClKind = *ClKindStr*

Specifies the clutch kind used in the user OpenXWD powertrain model. Following clutch kinds are supported:

- *Closed*: Permanently closed clutch
- *Open*: Permanently open clutch
- *Friction*: Friction clutch
- *Converter*: Converter clutch

PowerTrain.nPlanetGear = *value*

Specifies the number of planetary gears for powersplit hybrid. Default: 0.

PowerTrain.PlanetGear<pg>.Ratio = *value*

Specifies the ratio of the planetary gearbox between the ring and sun gear. Default: 2.6.

PowerTrain.nGearBoxM = *value*

Specifies the number of gearboxes for electric motors (without integrated starter generator motor). Default: 0.

PowerTrain.GearBox.GBKind = *GBKindStr*
PowerTrain.GearBoxM<gb>.GBKind = *GBKindStr*

Specifies the kind of the gearbox used in the user OpenXWD powertrain model.
Following kinds are supported:

- *NoGearBox*: No gearbox
- *Manual*: Manual gearbox
- *AutoWithManual*: Automatic gearbox with optional manual gear target (Manumatic)
- *AutoNoManual*: Automatic gearbox without optional manual gear target or continuously variable transmission (CVT)

PowerTrain.GearBox.ClKind = *ClKindStr*
PowerTrain.GearBoxM<gb>.ClKind = *ClKindStr*

Specifies the internal clutch kind in the gearbox used in the user OpenXWD powertrain model. Following clutch kinds are supported:

- *Closed*: Permanently closed clutch
- *Open*: Permanently open clutch
- *Friction*: Friction clutch
- *Converter*: Converter clutch

PowerTrain.GearBox.iForwardGears = *GearRatioList*
PowerTrain.GearBoxM<gb>.iForwardGears = *GearRatioList*

State the gearbox transmission ratio for every single forward gear.

PowerTrain.GearBox.iBackwardGears = *GearRatioList*
PowerTrain.GearBoxM<gb>.iBackwardGears = *GearRatioList*

State the gearbox transmission ratio for every single backward gear.

PowerTrain.nMotor = *value*

Specifies the total number of electric motors (without integrated starter generator motor). Default: 0.

PowerTrain.ISG.VoltageLevel = *LevelStr*
PowerTrain.Motor<m>.VoltageLevel = *LevelStr*

Specifies the voltage level (*LV*, *HV1* or *HV2*) of the power supply model to which the electric motor is connected.

PowerTrain.ISG.Ratio = *value*
PowerTrain.Motor<m>.Ratio = *value*

Ratio between electric motor shaft and driven shaft. Default: 1.0.

PowerTrain.ISG.Mot.rotv_max = *value*
PowerTrain.Motor<m>.Mot.rotv_max = *value*

Maximum electric motor rotation speed. Default: 8000rpm.

PowerTrain.ISG.Gen.rotv_max = *value*
PowerTrain.Motor<m>.Gen.rotv_max = *value*

Maximum electric generator rotation speed. Default: 8000rpm.

PowerTrain.ISG.Mot.TrqMap: *Table*
PowerTrain.Motor<m>.Mot.TrqMap: *Table*

One dimensional characteristic mapping for the maximum motor torque.

Syntax Infofile table mapping with 2 columns
<motor speed> [rpm] <motor torque> [Nm]

Example
```
PowerTrain.Motor.Mot.TrqMap:
 0.0    210
 682    210
1437    100
...
8000   18
```

PowerTrain.ISG.Gen.TrqMap: *Table*
PowerTrain.Motor<m>.Gen.TrqMap: *Table*

One dimensional characteristic mapping for the maximum motor/generator torque.

Syntax Infofile table mapping with 2 columns
<generator speed> [rpm] <generator torque> [Nm]

Example
```
PowerTrain.Motor.Gen.TrqMap:
 0.0    0
 500    210
1437    100
...
8000   18
```

PowerTrain.DL.iDiff_mean =*value*

Specifies the driveline mean differential ratio. Default: 3.0.

PowerTrain.DL.nDriveSource = *value*

Specifies the number of possible drive sources to the driveline. Default: 1.

PowerTrain.DL.DriveSourcePos<ds> = *PosStr*

Specifies on which position the drive source *ds* is applied. Possible positions are:

- *NoPosition*: No position specified
- *Center*: Center differential
- *Front*: Front differential
- *Front2*: Second front differential

- *Rear*: Rear differential
- *Rear2*: Second rear differential
- *FL*: Wheel FL
- *FR*: Wheel FR
- *RL*: Wheel RL
- *RR*: Wheel RR
- *RL2*: Wheel RL2
- *RR2*: Wheel RR2
- *FL2*: Wheel FL2
- *FR2*: Wheel FR2

PowerTrain.PowerSupply.BattLV.Active = *bool*
PowerTrain.PowerSupply.BattHV.Active = *bool*

Specifies if the low and high voltage batteries are implemented in the power supply model. Default: 1.

Depending on if the low and high voltage batteries are included, the following corresponding low or high voltage battery parameters are used:

PowerTrain.PowerSupply.BattLV.Capacity = *value*
PowerTrain.PowerSupply.BattHV.Capacity = *value*

Total capacity of the voltage battery. Default: 10Ah.

PowerTrain.PowerSupply.BattLV.Voltage_oc = *value*
PowerTrain.PowerSupply.BattHV.Voltage_oc = *value*

Battery open circuit (idle) voltage. Default: 12V.

PowerTrain.PowerSupply.BattLV.SOC_min = *value*
PowerTrain.PowerSupply.BattHV.SOC_min = *value*

Battery minimum admitted state of charge. Battery should not be discharged below this value. Default: 10%.

PowerTrain.PowerSupply.BattLV.SOC_max = *value*
PowerTrain.PowerSupply.BattHV.SOC_max = *value*

Battery maximum admitted state of charge. Battery should not be charged above this value. Default: 90%.

Chapter 16
Brake

16.1 Overview

The brake module has the role to calculate the current brake torque (wheel and park brake) for each wheel. These torques are supported at the wheel carrier and are used for the integration of the degree of freedom for wheel speed inside the powertrain module.

Using a hybrid powertrain with electric motors for regenerative braking, the brake module, or more precisely the brake control, communicates to the powertrain control the target regenerative braking torque at each wheel. It depends on the maximum possible regenerative braking torque of the motor(s) and the repartition strategy. Additionally the brake control pilots the brake system based on these current regenerative braking torques in order to achieve the desired total braking torque at each wheel. In combination with an ESP ECU it is possible to set a target drive source torque to powertrain control. All torques are transfered as absolute values.

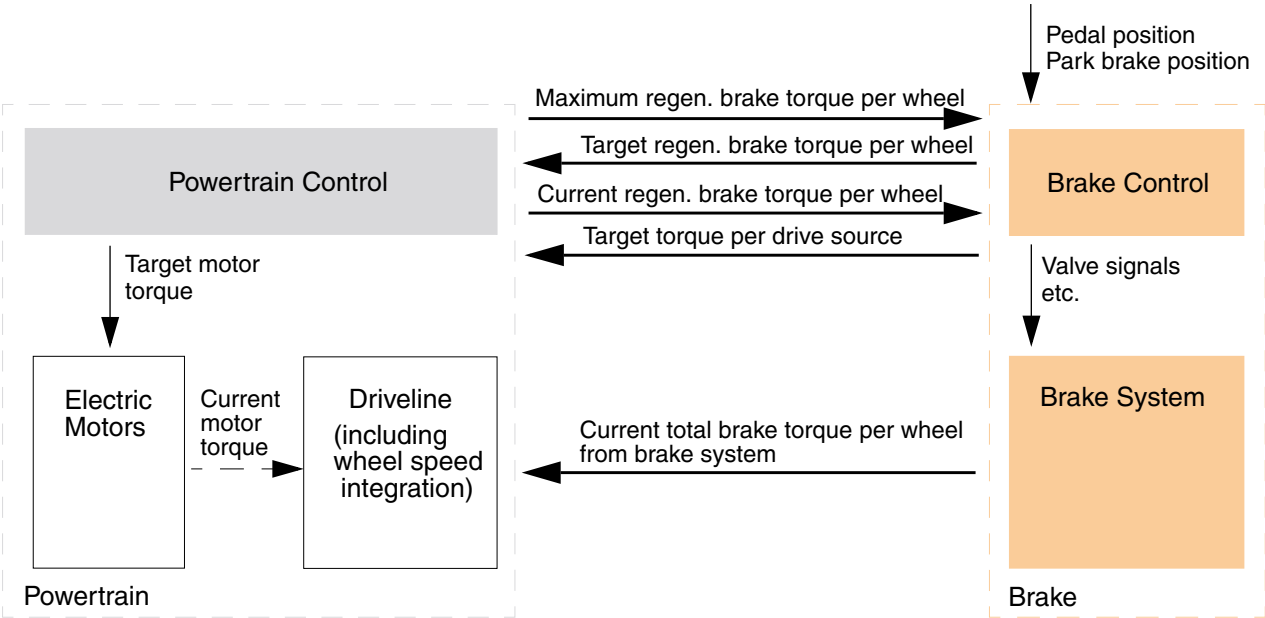


Figure 16.1: Communication between powertrain and brake

The parameters explained here apply for *all* brake models.

Brake.Kind = *KindStr VersionId*

Selection of brake model to use. The brake components library provides the following models:

ModelName	KindStr	Description
Hydraulic	Hyd	Hydraulic model, consisting of hydraulic control and brake system (see section 16.3 'Brake model "Hydraulic"')

Example `Brake.Kind = Hyd 1`

Brake.FName = *FName*

This references a brake model from a file including all other brake parameters (not listed here). The file is read from the subdirectory `Data/Misc`. The model type is determined from the `FileIdent` parameter, which is in the first line of the external file (`FileIdent = CarMaker-Brake-Hyd 1`).

If `Brake.FName` is specified, the parameter `Brake.Kind` is not taken into account.

Example `Brake.FName = MyBrake_DemoParam`

Brake.Torque.Amplify = *ValueList*

Multiplies the total brake torque of the wheel (including the park brake torque), which has been calculated by the brake model with this factor [-].
Four factors in the order front left, front right, rear left, rear right.

Example `Brake.Torque.Amplify = 1.0 1.0 1.0 1.0`

16.2 Brake Interface

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tBrakeCfgIF*:

Input Variable	Unit	Description
nWheels		Number of vehicle wheels
VhclClassId		Vehicle class identification number: 1: Car 2: MCycle 3: Truck 4: First trailer 5: Second trailer

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tBrakeIF*:

Input Variable	Unit	Description
Pedal		Brake pedal position
Park		Park brake position
T_env	K	Temperature of the environment
Trq_Reg[*wh*]	Nm	Estimated current regenerative brake torque at wheel *wh* (absolute value)
Trq_Reg_max[*wh*]	Nm	Estimated maximum possible regenerative brake torque at wheel *wh* (absolute value)

Output Variable	Unit	Description
Trq_WB[*wh*]	Nm	Wheel brake torque at wheel *wh* (absolute value)
Trq_PB[*wh*]	Nm	Parking brake torque at wheel *wh* (absolute value)
Trq_Reg_trg[*wh*]	Nm	Target regenerative braking torque at wheel *wh* (absolute value) to powertrain control
Trq_DriveSrc_trg[*ds*]	Nm	Target torque at drive source *ds* to powertrain control (e.g. by ESP ECU)

The Braking Concept

DrivMan and driver modules (also VehicleControl) are thinking normalized, 0..1. They do not know any absolute forces. Therefore, the vehicle deceleration is a “linear” function of *DrivMan.Brake* activity.

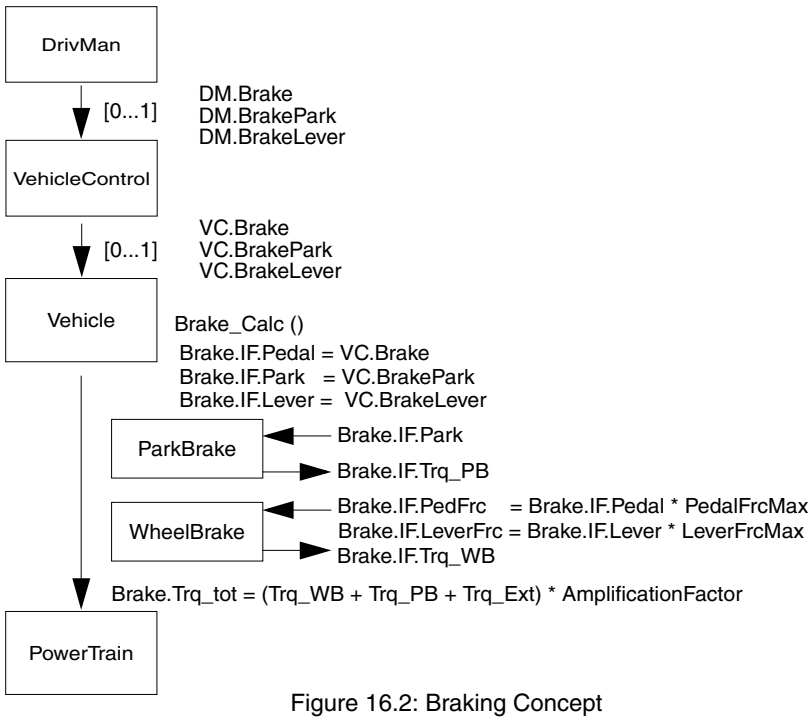


Figure 16.2: Braking Concept

Brake model "User C-code / Simulink Plug-in / FMU"

Please note:
The following output variable has to be set to -99999 (= "not set") if the signal is not provided by the Brake model. For Simulink Plug-ins the signal has to be set explicitly to -99999 else it will be set to 0!

Output Variable	Unit	Description
Trq_DriveSrc_trg[*ds*]	Nm	Target torque at drive source *ds* to powertrain control (e.g. by ESP ECU)

16.2.1 User Accessible Quantities for Brake

Please refer to section 24.11.1 'Brake - General'.

16.3 Brake model "Hydraulic"

The brake model *Hydraulic* consists of two parts: a hydraulic brake system and a corresponding brake control, which could also be a real brake ECU (using the IO module). Figure 16.1 shows the communication between both brake parts and between the brake and powertrain module. In the following the both parts are explained more in detail.

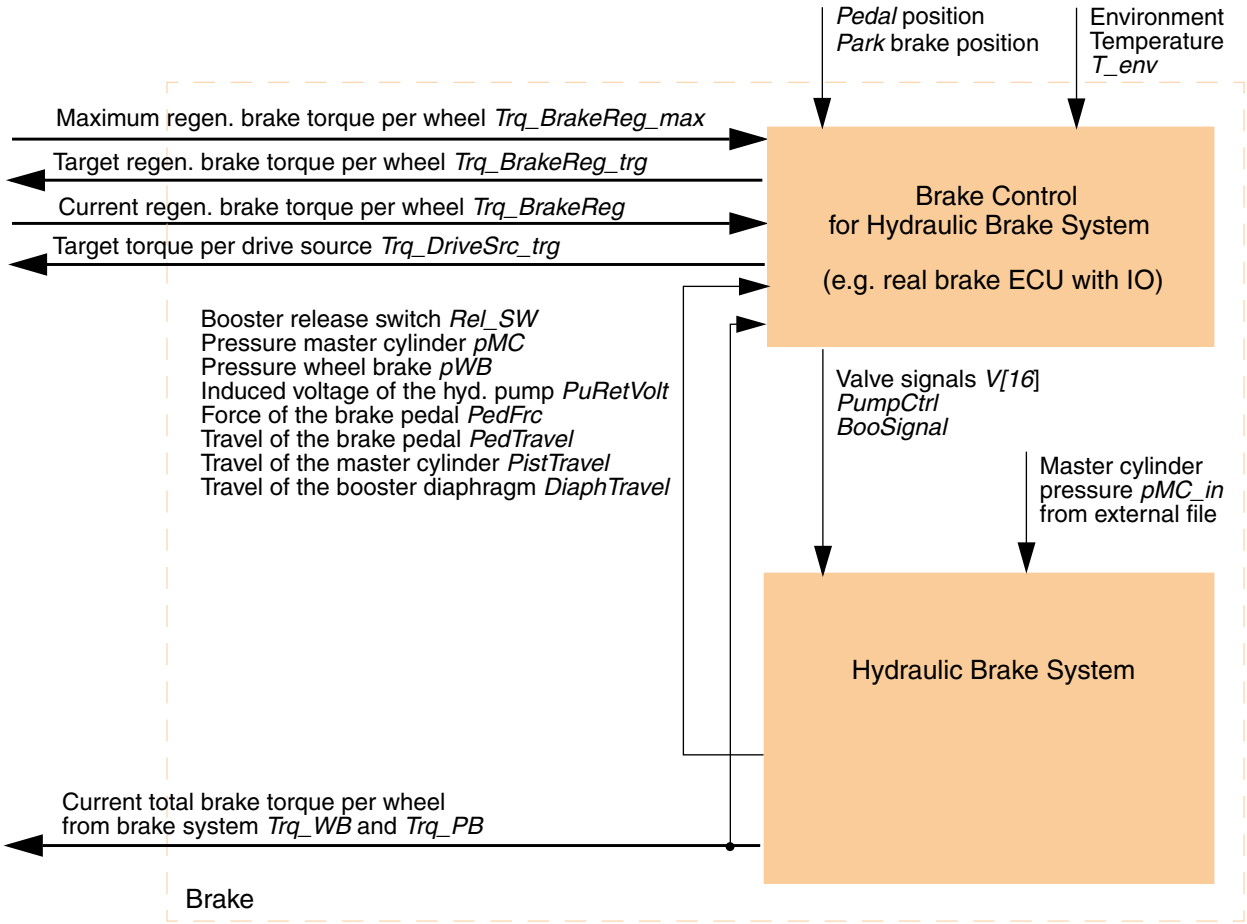


Figure 16.3: Structure of the brake model "Hydraulic"

16.3.1 Hydraulic Brake Control

Brake.Control.Kind = *KindStr VersionId*

Selection of the hydraulic brake control model to use. The brake components library provides the following models:

ModelName	KindStr	Description
Hydraulic Basic Control	HydBasic	Simple brake controller for hydraulic brake system with two strategies for torque repartition for regenerative braking
HIL	HydHIL	Model allows using a real ECU with IO

Example `Brake.Control.Kind = HydBasic 1`

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tHydBrakeCU_CfgIF*:

Input Variable	Unit	Description
nWheels		Number of vehicle wheels
VhclClassId		Vehicle class identification number: 1=Car, 2=MCycle, 3=Truck, 4=Trailer, 5=Trailer2
Kind		Brake model kind: 1=hydraulic brake, 2=pneumatic brake
KindStr		String of the brake model kind
Trq_stat[*wh*]	Nm	1D-Lookup table for static brake torque for wheel *wh* as function of pedal position

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tHydBrakeCU_IF*:

Input Variable	Unit	Description
Pedal		Brake pedal position
Park		Park brake position
T_env	K	Temperature of the environment
Trq_Reg[*wh*]	Nm	Estimated current regenerative brake torque at wheel *wh* (absolute value)
Trq_Reg_max[*wh*]	Nm	Estimated maximum possible regenerative brake torque at wheel *wh* (absolute value)
Trq_WB[*wh*]	Nm	Wheel brake torque at wheel *wh* (absolute value)
Trq_PB[*wh*]	Nm	Parking brake torque at wheel *wh* (absolute value)
Rel_SW		Booster release switch
pMC	bar	Pressure actuation unit (master cylinder)
pWB[*wh*]	bar	Pressure at wheel brake *wh*
PuRetVolt	V	Induced voltage of the hydraulic pump
PedFrc	N	Force on the brake pedal
PedTravel	m	Travel of the brake pedal
PistTravel	m	Travel of the master cylinder piston
DiaphTravel	m	Travel of the booster diaphragm

Output Variable	Unit	Description
Pedal		Brake pedal position
Park		Park brake position
Trq_Reg_trg[*wh*]	Nm	Target regenerative braking torque at wheel *wh* (absolute value) to powertrain control
Trq_DriveSrc_trg[*ds*]	Nm	Target torque at drive source *ds* to powertrain control
V[16]		Relative valve signals (0...1)
PumpCtrl		Pump control/activation (0...1)

Output Variable	Unit	Description
BooSignal		Booster input signal

Hydraulic Brake Control model "User C-code / Simulink Plug-in / FMU"

Please note:
The following output variable has to be set to -99999 (= "not set") if the signal is not provided by the Brake model. For Simulink Plug-ins the signal has to be set explicitly to -99999 else it will be set to 0!

Output Variable	Unit	Description
Trq_DriveSrc_trg[*ds*]	Nm	Target torque at drive source *ds* to powertrain control

Hydraulic Brake Control model "HydBasic"

With the brake control model *HydBasic* for hydraulic brake systems it is possible to brake with regenerative wheel brake torque in a hybrid powertrain. Figure 16.4 demonstrates the functionality of this model:

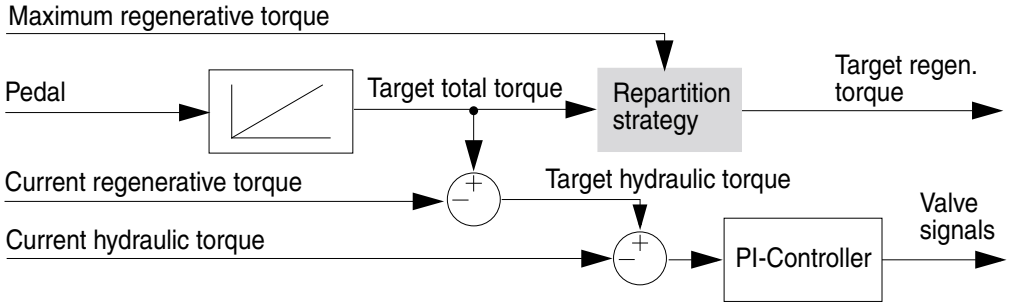


Figure 16.4: Functionality of the brake control model

With the current brake pedal position a target total brake torque for each wheel is determined. Depending on the repartition strategy (two variants are shown in Figure 16.5) a target regenerative wheel brake torque for the powertrain control is identified. It is limited by the maximum possible regenerative brake torque at the wheel (green area). The second

task of the model is to control the residual brake torque (magenta area) using the hydraulic brake system to get the desired total brake torque. For this the valve signals of the hydraulic brake model are controller by a PI-controller to obtain the target hydraulic brake torque.

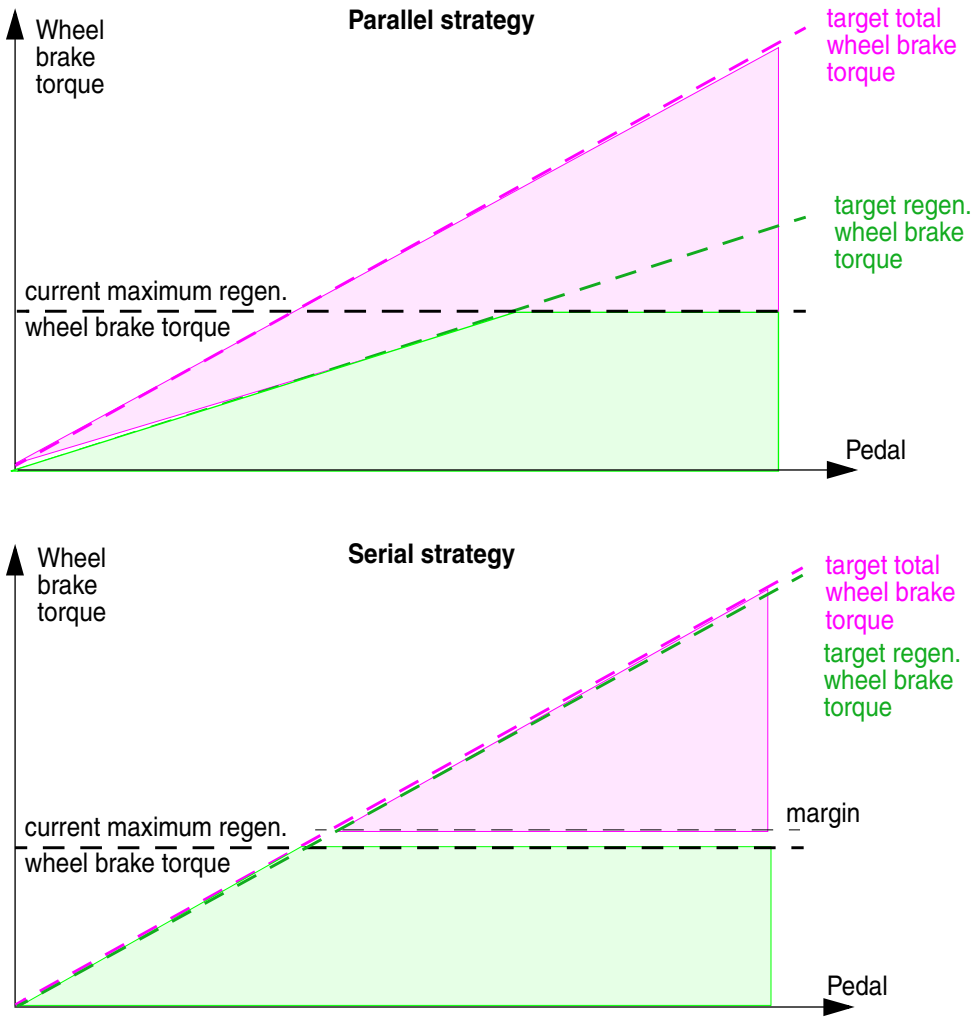


Figure 16.5: Brake torque repartition strategy

Brake.Control.Mode = *ModeStr*

Specifies the repartition mode of the brake control model. Possible modes are:

- *NoController*: No controller used, no calculation of regenerative brake torque (default)
- *Parallel*: Parallel repartition: the regenerative and hydraulic brake torques are applied at the same time, given by a constant ratio
- *Serial*: Serial repartition: as long as the target total brake torque is smaller than the maximum possible regenerative brake torque, the brake torque is generated by the powertrain only. If the target total torque exceeds a margin torque, the hydraulic brake torque supports the powertrain.

Brake.Control.Ratio = *value*

Specifies the ratio of the target regenerative brake torque to total brake torque. Used only for the parallel repartition mode. Default: 0.5.

Brake.Control.Margin = *value*

If the difference between the target brake torque and the maximum possible regenerative brake torques exceeds the margin value, then the hydraulic brake system is activated to support the regenerative brake torques (see Figure 16.5). Used only for the serial repartition mode. Default: 10 Nm.

Brake.Control.TrqCtrl.p = *value*

Specifies the proportional value of the PI-torque controller for the brake system valve signals. Default: 0.01.

Brake.Control.TrqCtrl.i = *value*

Specifies the integral value of the PI-torque controller for the brake system valve signals. Default: 0.01.

16.3.2 Hydraulic Brake System

Overview

CarMaker offers for every purpose an appropriate hydraulic brake system model. There are different interests for using a specific brake system model.

A simple brake model is sufficient, if the main interest of investigation is not the brake system itself and no brake ECU is used. It is good for driving maneuvers with simple braking tasks.

If the interest of investigation is the hydraulic brake system itself or if a brake ECU has to be satisfied, there is a high resolution brake model available. The aim is to represent the real braking system as closely as possible in combination with real time computing constraints.

Brake.System.Kind = ***KindStr VersionId***

Selection of hydraulic brake system model to use. The brake components library provides the following models:

ModelName	KindStr	Description
Pressure Distribution	PresDistrib	Simple brake system model, for details see section 16.4 'Hydraulic Brake System "Pressure Distribution"'
HydESP	HydESP	High resolution brake system model, for details see section 16.5 'Hydraulic Brake System "HydESP"'

Example `Brake.System.Kind = HydESP 1`

Brake.System.FName = ***FName***

This references a hydraulic brake system model from a file including all other (not listed here) brake parameters. The file is read from the subdirectory `Data/Misc`. The model type is determined from the `FileIdent` parameter, which is in the first line of the external file (`FileIdent = CarMaker-HydBrakeSystem-HydESP 3`).

If `Brake.System.FName` is specified, the parameter `Brake.System.Kind` is not taken into account.

Example `Brake.System.FName = Hyd_DemoParam`

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tHydBrakeCfgIF*:

Input Variable	Unit	Description
nWheels		Number of vehicle wheels
VhclClassId		Vehicle class identification number: 1: Car 2: MCycle 3: Truck 4: First trailer 5: Second trailer
Kind		Brake model kind: 1: hydraulic brake 2: pneumatic brake
KindStr		String of the brake model kind

Output Variable	Unit	Description
Trq_stat[*wh*]	Nm	1D-Lookup table for static brake torque for wheel *wh* as function of pedal position
Ped2pMC	bar	1D-Lookup table for static master cylinder pressure as function of pedal position

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tHydBrakeIF*:

Input Variable	Unit	Description
Pedal		Brake pedal position
Park		Park brake position
T_env	K	Temperature of the environment
V[16]		Relative valve signals (0...1)
PumpCtrl		Pump control/activation (0...1)
BooSignal		Booster input signal
Use_pMCInput	bool	Flag to use pressure for the master cylinder from external file (Input From File mechanism)
pMC_in	bar	Input from external file for the master cylinder pressure

Output Variable	Unit	Description
Trq_WB[*wh*]	Nm	Wheel brake torque at wheel *wh* (absolute value)
Trq_PB[*wh*]	Nm	Parking brake torque at wheel *wh* (absolute value)
Rel_SW		Booster release switch
pMC	bar	Pressure actuation unit (master cylinder)
pWB[*wh*]	bar	Pressure at wheel brake *wh*
PuRetVolt	V	Induced voltage of the hydraulic pump
PedFrc	N	Force on the brake pedal

Output Variable	Unit	Description
PedTravel	m	Travel of the brake pedal
PistTravel	m	Travel of the master cylinder piston
DiaphTravel	m	Travel of the booster diaphragm
OptDblIO OptLngIO OptPtrIO		Optional IO

Hydraulic Brake System model "User C-code / Simulink Plug-in / FMU"

The following parameters are required for the initialization of the output variables in the interface struct *tHydBrakeCfgIF*:

Brake.System.Wheel.<wh>.Trq_stat: *Table*

One dimensional characteristic mapping for the static brake torque at wheel <wh=0...7> (absolute value) depending on the brake pedal position (0...1).

Syntax
Infofile table mapping with 2 columns
<brake pedal position 0...1> [] <static brake torque, absolute value> [Nm]

Example

```
Brake.System.Wheel.0.Trq_stat:
 0.0    0
 0.1    150
 0.2    300
 ...
 1.0   1500
```

Brake.System.Ped2pMC: *Table*

One dimensional characteristic mapping for the static master cylinder pressure depending on the brake pedal position (0...1).

Syntax
Infofile table mapping with 2 columns
<brake pedal position 0...1> [] <master cylinder pressure> [bar]

Example

```
Brake.System.Ped2pMC:
 0.0    0
 0.1    10
 0.2    20
 ...
 1.0     100
```

16.3.3 User Accessible Quantities for Brake model "Hydraulic"

Please refer to section 24.11.2 'Brake - Hydraulic'.

16.4 Hydraulic Brake System "Pressure Distribution"

16.4.1 Overview

This is a simple hydraulic brake system model which acts like a conventional one circuit brake.

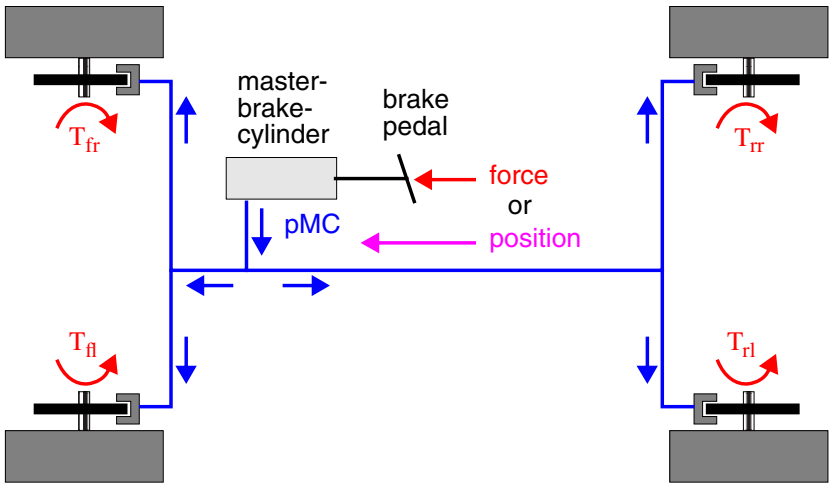


Figure 16.6: Principle of brake model ‘Pressure Distribution’

A master brake pressure is build proportional to the input value of the brake pedal.

$$p_{MC} = PF2pMC \cdot F_{ped} \quad \text{(EQ 212)}$$

or

$$p_{MC} = PP2pMC \cdot x_{ped} \quad \text{(EQ 213)}$$

The braking torque for each individual wheel is calculated by

$$T_i = p_{MC} \cdot pWB2Trq_i \quad \text{with } i = \text{fl, fr, rl, rr} \quad \text{(EQ 214)}$$

p_{MC}	pressure of master brake cylinder
F_{ped}	brake pedal force
x_{ped}	brake pedal position
T_i	brake torque at wheel i
$PF2pMC$	conversion factor from brake pedal force to master brake cylinder pressure
$PP2pMC$	conversion factor from brake pedal position to master brake cylinder pressure
$pWB2Trq_i$	conversion factor from master brake cylinder pressure to wheel brake torque

16.4.2 Hydraulic Brake System "Pressure Distribution" Parameters

Brake.System.pMC_based_on

This parameter specifies how the master cylinder brake pressure is determined.

pMC_Based_on	Description
PedalAct	Master brake pressure depends on brake pedal activation (Default).
PedalForce	Master brake pressure depends on brake pedal force.

The brake pressure in the master cylinder is proportional to either pedal activation or pedal force.

Example `Brake.System.pMC_based_on = PedalAct`

Brake.System.PedalAct2pMC

This parameter is valid if 'PedalAct' is selected with `Brake.System.pMC_Based_on`. This factor determines the master brake cylinder pressure according to (EQ 213) [bar].

Brake.System.Pedal2PedalFrc

This parameter is valid if 'PedalForce' is selected with `Brake.System.pMC_Based_on`. This factor determines the pedal force as a function of the pedal position [N].

Brake.System.PedalForce2pMC

This parameter is valid if 'PedalForce' is selected with `Brake.System.pMC_Based_on`. This factor determines the master brake cylinder pressure according to (EQ 212) [bar/N].

Brake.System.pWB2Trq

Ratio of the wheel brake cylinder pressure to brake torque, see (EQ 214) [Nm/bar]. Specify 4 values for fl, fr, rl, rr wheel.

Example `Brake.System.pWB2Trq = 16.0  16.0  7.0  7.0`

Brake.System.tResp

Response time of the braking system. Corresponds to the elapsed time between the pedal activation and the moment when the braking force takes effect. Unit: [s]. Default: 0.005.

Brake.System.tBuildUp

Build-up time of the braking system. Corresponds to the elapsed time between the moment when the braking force takes effect and the moment when the braking force reaches 75% of the desired braking force. Unit: [s]. Default: 0.08.
Value 0 has a special function: response time and build-up time are set to zero: no time delay.

Parking Brake

CarMaker distinguishes between brake and parking brake. The parking brake model is contained in the "Pressure Distribution" brake model.

Brake.System.Park.Torque_max = *ValueList*

Maximum parking brake torque at each wheel [Nm]. Four values in the order front left, front right, rear left, rear right.

Example

```
Brake.System.Park.Torque_max =  0  0  1000  1000
```

In this example the parking brake acts only on the rear wheels.

16.5 Hydraulic Brake System "HydESP"

16.5.1 Overview

Structure of the hydraulic braking system

A standard car hydraulic braking system consists of two brake circuits, each one applying two wheel brakes. Also this model can be used with two different brake-circuit configurations:

In the diagonal (X-pattern) system, each brake circuit applies on a front wheel brake and a rear wheel brake on the opposite side. In the system with separate circuits for the front and rear axle (II-pattern), each circuit applies the brakes on one axle.

The two circuits are called *primary* and *secondary circuit*. In reality as well as in the model the two circuits are identical.

The following picture shows a brake system with ESP. Only the primary circuit and only one wheel brake is shown.

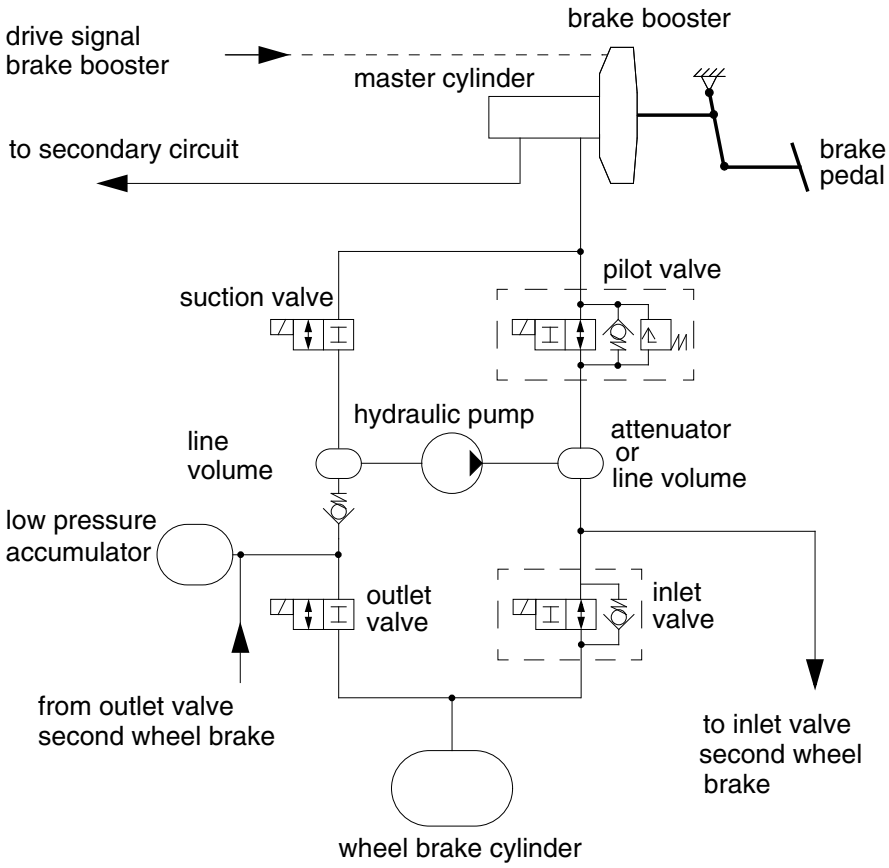


Figure 16.7: ESP Braking System

Interface of the model

The HydESP model has interfaces to signals of the driver, the brake ECU and the vehicle. The following lists name the input and output used and generated by the brake model.

Inputs

- normalized brake pedal actuation [0..1]
- normalized park brake actuation [0..1]
- drive signals for the solenoid valves: digital or analog [0..1]
- pump on/off (digital)
- brake booster drive signal

Outputs

- pressure master cylinder
- pressures wheel brake cylinders
- brake torques
- park brake torques
- return voltage of the hydraulic pump
- state of the booster release switch
- travel of the master cylinder piston
- travel of the brake pedal

Functional Description

Figure 16.7 shows the simplified model of a ESP controlled brake system. The following is a brief description of such a system and how it features are modeled.

The brake force applied to the brake pedal by the driver is mechanically transformed into a input force of the brake booster. The brake booster itself amplifies the input force and passes its output to the pistons of the master brake cylinder. There the piston rod force is translated into a master cylinder pressure.

The master cylinder has two hydraulic connectors one for each brake circuit. It is assumed that the master cylinder pressure is equal for both brake circuits.

The hydraulic part is modeled as a system of alternating volume and connection elements (connecting lines including valves, hydraulic pumps, etc.). This means that from every volume element there is a flow of brake fluid through connection elements to another volume element (depicted in Figure 16.8).

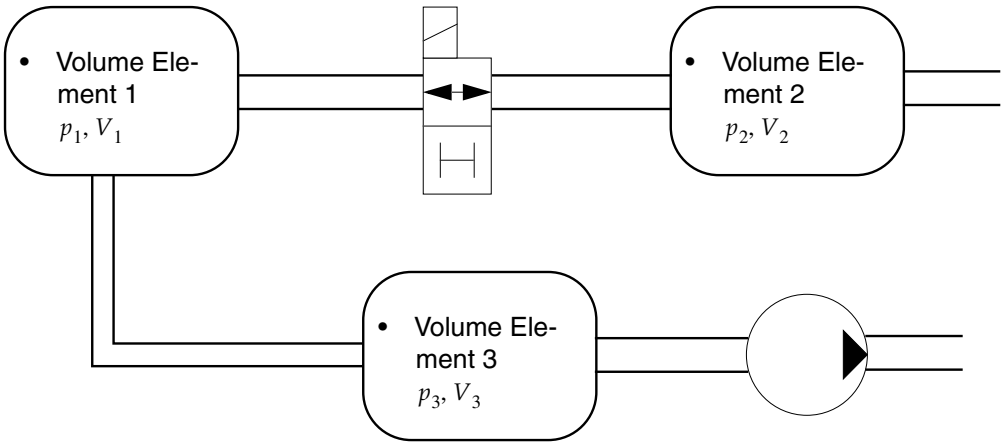


Figure 16.8: Hydraulic model with alternating volume and connection elements

Each volume element is characterized with the state variables pressure and volume. The system tries to equalize pressure differences with a flow from the high pressure to the low pressure volume element. The flow rate is limited by the hydraulic resistance which is a attribute of the connection element. The hydraulic resistance is a combination of the resistance of the connecting line and the current resistance of the valves (which often is a function of time and/or pressure difference). In case of a hydraulic pump this is reversed and the pump transports fluid from the low pressure side to the high pressure side.

As a counterpart to the master brake cylinder the wheel brake cylinder transforms its current pressure into a brake lining force. This force is transformed into a brake torque by the wheel brake itself.

The goal of the system of valves, pumps, connection lines and volume elements in case of a ESP system is to control the brake torque of each single wheel by regulating the pressure of its wheel brake cylinder. This is done by opening and closing the inlet and outlet valves for a certain amount of time to increase or decrease the brake pressure. To hold the current pressure inlet and outlet valves remain closed at the same time.

The hydraulic pump is used to pump fluid back from the low pressure accumulator (where brake fluid is temporarily stored) to the reservoir of the master brake cylinder. In special operation modes of the system when the driver is not actuating the brake pedal the pump is used to generate pressure for autonomous brake interventions. For those operation modes of a ESP system two more controlled valves are needed for each brake circuit (named suction and pilot valves in Figure 16.7). See technical literature for more information how the suction and pilot valves are used and how specific actions are operated.

ABS systems have basically the same structure as ESP systems but the suction and pilot valves are omitted. They are not needed because ABS systems do not perform autonomous brake interventions.

16.5.2 Brake circuit configuration

CircuitConfig

Optional. This parameter specifies the brake circuit configuration X- vs. II-pattern. (X-pattern = diagonal split, II-pattern = one circuit for one axle). Default: X-pattern.

CircuitConfig	Brake Circuit Index	Brake Circuit	Description
X	0	FR/RL	Diagonal split.
	1	FL/RR	
II	0	FL/FR	Parallel (front/rear) split.
	1	RL/RR	

Example `CircuitConfig = X`

Brake pedal

The brake force applied to the brake pedal by the driver is mechanically transformed into a piston rod input force of the brake booster.

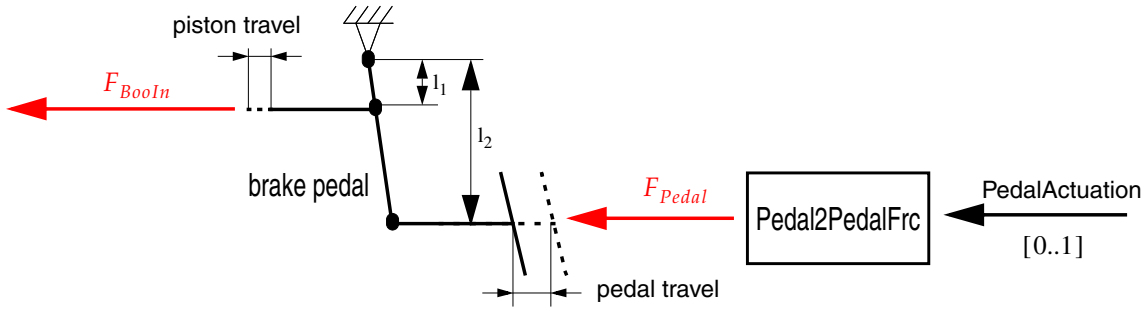


Figure 16.9: Mechanical amplification of drivers brake force

$$F_{BooIn} = ratio \cdot F_{Pedal} \quad \textbf{(EQ 215)}$$

Pedal.ratio

The pedal ratio amplifies the force of the brake pedal [-]. The resulting force is the input of the brake system.

Example `Pedal.ratio = 3.0`

Pedal2PedalFrc

Optional. This parameters defines the relation between driver brake pedal actuation and brake pedal force F_{Pedal} . Therefore a linear equation is used:

$$F_{\mathrm{Pedal}} = \mathrm{PedalActuation} \cdot \mathrm{Pedal2PedalFrc} \quad \textbf{(EQ 216)}$$

The force [N] defines the maximum brake force with full pedal actuation [1]. Default: 300 N.

Example `Pedal2PedalFrc = 500`

16.5.3 Brake booster

The brake booster amplifies the input force at the piston rod proportionally up to a certain limit. Above this limit there is no more amplification of force.

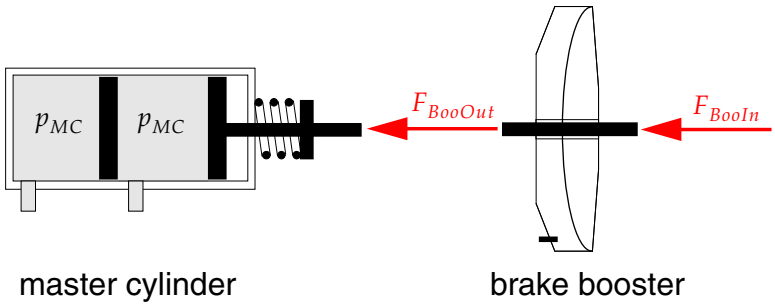


Figure 16.10: Amplification of brake actuation force by the brake booster.

The output force of the brake booster F_{BooOut} equals a pressure of the master cylinder. This relation can be calculated with (EQ 220) of the master brake cylinder.

Many parameters of the brake booster expect master cylinder pressures as input values instead of booster output forces. See the particular description of these parameters.

Figure 16.11 shows the relation between the input force of the brake booster and the corresponding master cylinder pressure. Before any pressure is build up the precharge force F_0 of the springs in the master cylinder has to be overcome. The output force of the booster is then calculated as follows:

$$F_{BooOut} = ratio \cdot F_{BooIn} \quad \text{(EQ 217)}$$

The amplification ratio is higher than one.

When a specific output force (or an corresponding master cylinder) is exceeded the amplification ratio becomes one for any further increase of input force. This pressure is called the booster run out pressure. The corresponding force can be calculated with (EQ 220).

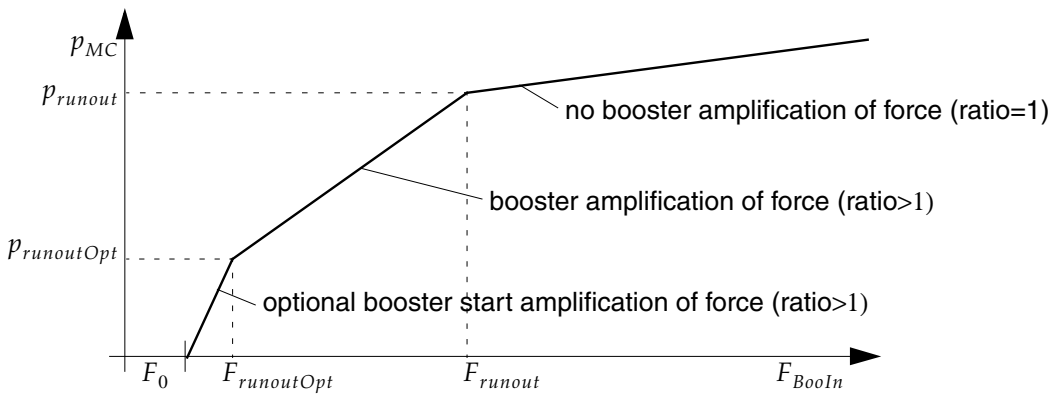


Figure 16.11: Brake booster output pressure characteristic

Actually it is possible to specify two different amplifications and corresponding run out pressures (see Figure 16.11). The first amplification and run out pressure parameters are optional, they are mainly thought to allow a higher amplification at the beginning of the pressure build up.

The following brake booster types are provided with this hydraulic model:

Boo.type

Optional. Boo.type determines the type of booster or precharge pump used. Some brake systems additionally use an electric drive of the booster in order to create a brake pressure even if there is no pedal force (e.g. brake assist, ACC). See booster models `TargetPressure` and `PressureGradient`.

Boo.type	Description
none	*Default*: No brake booster is used. (Amplification ratio is always one).
Mechanical	Classic mechanical brake booster.
TargetPressure	Booster with pressure proportional to input signal or precharge pump.
PressureGradient	Booster with pressure gradient dp/dt = f(input signal)

Brake Booster “Mechanical”

See Figure 16.11: for explanation of this booster model.

Boo.ampli

Amplification ratio when the brake booster is in amplification range [-]. See (EQ 217).

Example `Boo.ampli = 7`

Boo.runOut

Specifies the booster run-out pressure [bar]. Above the run-out pressure the booster amplification is one (no amplification).

Example `Boo.runOut = 90`

Boo.ampliOpt

Optional. If this parameter is available the booster amplification is two-staged. This parameter then specifies the amplification of the first stage, whereas Boo.ampli specifies the amplification of the second stage.
Boo.runOutOpt must be set as well. (see Figure 16.11:).
Default: Boo.ampli.

Example `Boo.ampliOpt = 15`

Boo.runOutOpt

Optional. Specifies the booster run-out pressure [bar] of the first stage. Above this run-out pressure the booster amplification is specified by Boo.ampli.
Requires: Boo.runOutOpt < Boo.runOut
Default: 0.0.

Example `Boo.runOutOpt = 18`

Brake Booster “TargetPressure”

“TargetPressure” is a model of a controlled system consisting of the mechanical booster and the simulated dynamic behavior of a booster-ECU. Internally this model distinguishes two forces:

- A “interactive” booster force by amplification of the drivers input force like the booster model “mechanical”.
- A “automatic” booster force which is initiated by the standardized drive signal. The drive signal which is a input quantity of the HydESP brake interface has to be controlled by user code (can be a simulated or real ECU). The output force is proportional to the drive signal (0..1) without any driver interaction necessary for this.

  The dynamic behavior is calculated within this model. The dynamics of the force increase is modeled by a delay and a exponential saturation function.

  Additionally a force threshold value for the release switch must be given. The release switch is operated when the piston rod force (driver interaction) exceeds the given threshold force.

The effective booster force calculates as follows:

$$F_{\mathrm{BooOut}} = max(F_{\mathrm{Boo\ interactive}}, F_{\mathrm{Boo\ automatic}}) \qquad \textbf{(EQ 218)}$$

For the “interactive” case see Figure 16.11: , the “automatic” case is shown in the following illustration:

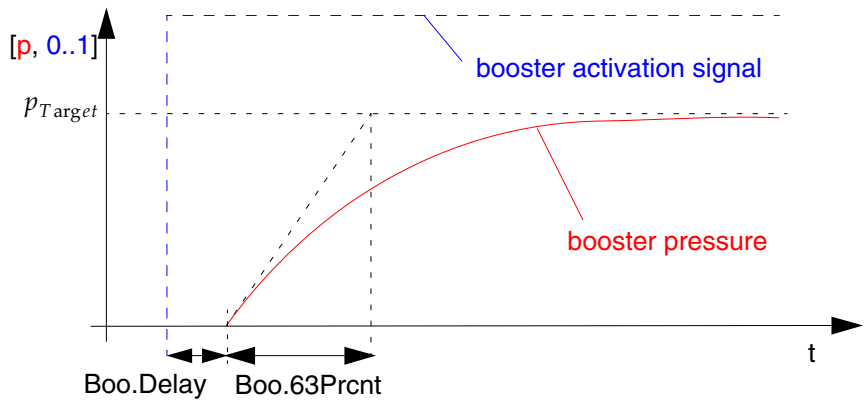


Figure 16.12: Pressure rise controlled by booster ECU (case “automatic” booster force)

Boo.ampli

Amplification ratio when the brake booster is in amplification range [-]. See (EQ 217).

Example `Boo.ampli = 7`

Boo.runOut

Specifies the booster run-out pressure [bar]. Above the run-out pressure the booster amplification is one (no amplification).

Example `Boo.runOut = 90`

Boo.ampliOpt

Optional. If this parameter is available the booster amplification is two-staged. This parameter then specifies the amplification of the first stage, whereas Boo.ampli specifies the amplification of the second stage.
Boo.runOutOpt must be set as well. (see Figure 16.11:).
Default: Boo.ampli.

Example `Boo.ampliOpt = 15`

Boo.runOutOpt

Optional. Specifies the booster run-out pressure [bar] of the first stage. Above this run-out pressure the booster amplification is specified by Boo.ampli.
Requires: Boo.runOutOpt < Boo.runOut
Default: 0.0.

Example `Boo.runOutOpt = 18`

Boo.delay

Delay between signal and pressure rise [s].

Example `Boo.delay = 0.01`

Boo.63Prcnt

Time constant for pressure rise [s]. After $t = \text{Boo.Delay} + \text{Boo.63Pcnt}$, booster pressure has reached 63% of the target value.

Example `Boo.63Prcnt = 0.01`

Boo.sign2press

This parameter determines the output (target) pressure when booster ECU is active [bar/1]. The range of the drive signal is 0..1.

$$p_{Target} = \text{Boo.sign2press} \cdot \text{DriveSignal}$$

Example `Boo.sign2press = 1`

Boo.relF

Pedal force applied by driver to open the release switch [N].

Example `Boo.relF = 10`

Brake Booster “PressureGradient”

“PressureGradient” is a booster model similar to the model “Target Pressure”. The difference is that the drive signal is interpreted in a different way. Like the booster model “Target Pressure” two forces are distinguished:

- A “interactive” booster force by amplification of the drivers input force like the booster model “mechanical”.
- A “automatic” booster force is initiated by the standardized drive signal of the booster ECU which is a input quantity of the HydESP brake interface.

  Here the gradient of pressure increase/decrease is a function of the activation signal. The pressure can not exceed a maximum pressure.

  In this booster model the release switch is operated when the “interactive” pedal force exceeds the force generated by the “automatic” booster force.

In case of an autonomous intervention parallel to a pedal force, the resulting pressure is the maximum of the two pressures:

$$F_{\text{Booster}} = \max(F_{\text{Booster manual}}, F_{\text{Booster ECU}}) \quad \textbf{(EQ 219)}$$

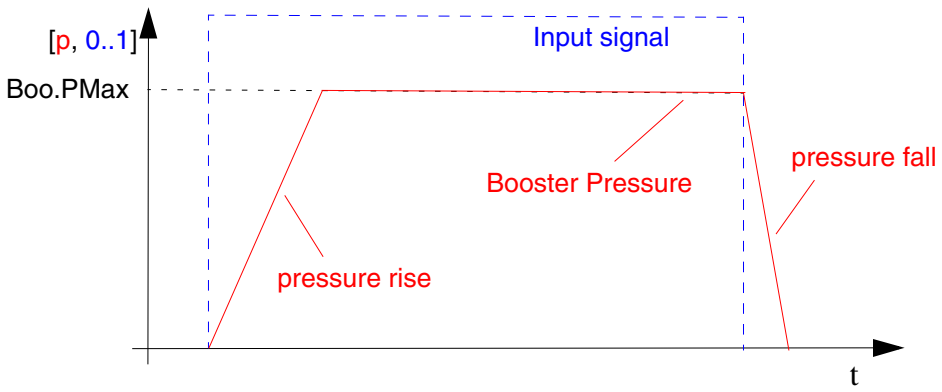


Figure 16.13: Pressure rise with booster model PressureGradient

Boo.ampli

Amplification ratio when the brake booster is in amplification range [-]. See (EQ 217).

Example `Boo.ampli = 7`

Boo.runOut

Specifies the booster run-out pressure [bar]. Above the run-out pressure the booster amplification is one (no amplification).

Example `Boo.runOut = 90`

Boo.ampliOpt

Optional. If this parameter is available the booster amplification is two-staged. This parameter then specifies the amplification of the first stage, whereas Boo.ampli specifies the amplification of the second stage.
Boo.runOutOpt must be set as well. (see Figure 16.11:).
Default: Boo.ampli.

Example `Boo.ampliOpt = 15`

Boo.runOutOpt

Optional. Specifies the booster run-out pressure [bar] of the first stage. Above this run-out pressure the booster amplification is specified by Boo.ampli.
Requires: Boo.runOutOpt < Boo.runOut
Default: 0.0.

Example `Boo.runOutOpt = 18`

Boo.pMax

Maximum pressure difference the booster is able to produce in “automatic” mode (this means booster pressure is controlled by the ECU) [bar].

Example `Boo.pMax = 90`

Boo.pGrad.mapping

Relation of booster activation signal [0..1] to gradient of pressure rise/fall [bar/s]. .

Syntax Infofile table mapping with 2 columns
<Act.-Signal> <Pressure rise/fall>

Example Boo.pGrad.mapping:

0.0	-1000.0
0.15	-1000.0
0.3	0.0
0.4	0.0
0.8	400.0
1.0	400.0
1.1	400.0

16.5.4 Master Cylinder

The master cylinder transforms the output force of the brake booster to a brake pressure.

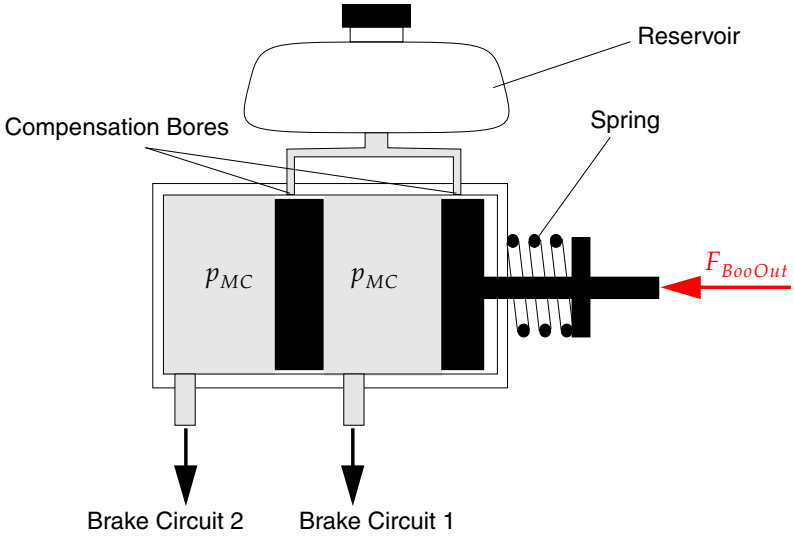


Figure 16.14: Master Brake Cylinder

$$p_{MC} = \frac{F_{BooOut} - F_{MC,0} - c_{MC} \cdot x_{MC}}{A_{MC}} \text{ for } x_{MC} > x_{CompBore} \quad \textbf{(EQ 220)}$$

with

p_{MC}	Master cylinder pressure
F_{BooOut}	Brake booster output force
A_{MC}	Area of master cylinder.
$F_{MC,0}$	Precharge of the springs in MC.
c_{MC}	Spring constant of the MC springs.
x_{MC}	Piston travel.
$x_{CompBore}$	Piston travel to close compensation bore.

The pressure generated in the primary circuit is equal to the one in the secondary circuit.

When the force is zero, the piston is at the backside stop. The compensation bores are open in this position and lead to a pressure equalization with the compensating reservoir (=atmospheric pressure). Once the force becomes superior to the precharge of the springs, the piston moves forward and closes the compensation bores. The whole brake system is now a closed system and the further movements of the piston are determined by the system's elasticity.

MC.area

Area of the piston [cm^2]

Example `MC.area = 4.5`

MC.xCompBore

Piston travel to close compensation bore [mm]

Example `MC.xCompBore = 2`

MC.springConst

$\frac{df}{dx}$ of the spring(s) $\left[\frac{\mathrm{N}}{\mathrm{m}}\right]$.

Example `MC.springConst = 1000`

MC.springLoad

Precharge of the spring(s) [N].

Example `MC.springLoad = 100`

16.5.5 Wheel brakes

The following calculation can be applied to disk or drum brakes because it is very general. Figure 16.15: explains the requested parameters.

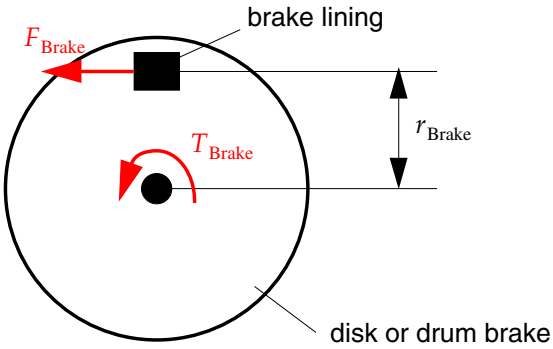


Figure 16.15: Schematic view of wheel brakes

The pressure in the brake cylinder is transformed into a brake torque as follows:

$$T_{Brake} = F_{Brake} \cdot r_{Brake} = p_{Brake} \cdot A_{BC} \cdot a_{Brake\ lining} \cdot r_{Brake} \quad \textbf{(EQ 221)}$$

with

p_{Brake}	Brake pressure
T_{Brake}	Brake torque.
A_{BC}	Area of brake cylinder.
$a_{Brake\ lining}$	Brake lining ratio.
F_{Brake}	Brake force acting at the brake linings.
r_{Brake}	Effective brake radius transforming force into torque.

The brake lining ratio describes the relation between brake pressure and resulting wheel brake torque. For disc brakes, its value usually is the coulomb number multiplied by 2. (factor 2 because there are brake linings on both sides of the disc).

Optionally, a characteristic can be given:

$$a_{Brake\ lining} = f(p_{Brake}) \quad \textbf{(EQ 222)}$$

The given pairs of values are interpolated linearly.

The brake radius is the effective radius to calculate the brake torque out of the brake force.

For the following parameters applies: ‘f’ means front , ‘r’ means rear, ‘fl’, ‘fr’ means front left/right and ‘rl’, ‘rr’ means rear left/right.

Pist_f.area
or alternatively:
Pist_fl.area
Pist_fr.area

Pist_r.area
or alternatively:
Pist_rl.area
Pist_rr.area

Total *effective* area of the *ensemble* of brake cylinders of one side (*outboard or inboard*) of brake pistons of a *single* brake [cm^2].

Example `Pist_f.area = 23.0`

Pist_f.rBrake
or alternatively:
Pist_fr.rBrake
Pist_fl.rBrake

Pist_r.rBrake
or alternatively:
Pist_rl.rBrake
Pist_rr.rBrake

Effective brake radius (Brake Force -> Brake Torque) [m].

Example `Pist_f.rBrake = 0.1`

Pist_fl.ratio
Pist_fr.ratio
Pist_rl.ratio
Pist_rr.ratio

Ratio of braking force to actuating force [-]. This parameter takes into account the influence of the internal *transmission ratio* of the brake as well as the Coulomb friction coefficient. Frequently: 2*Coulomb (factor 2 comes from inboard + outboard brake linings)
The ratio must be given for each brake of the four brakes.

Example `Pist_fl.ratio = 0.7`

Pist_fl.ratio.mapping
Pist_fr.ratio.mapping
Pist_rl.ratio.mapping
Pist_rr.ratio.mapping

Optional, instead of `Pist_<i>.ratio`. Ratio [-] of braking force to actuating force as a function of the applied brake pressure [bar]. .

Syntax

Infofile table mapping with 2 columns
<Brake pressure> <ratio>

Example

Pist_fl.ratio.mapping:

0	0.7
50	0.71
100	0.72
150	0.74
200	0.8

16.5.6 Volume elements in general

Volume elements have inputs and outputs. The resulting liquid volume leads to a certain pressure in the volume element. This relation is described by a function:

$$p = f(\text{absorbed liquid volume}) . \qquad \textbf{(EQ 223)}$$

By definition, *the liquid volume of the filled elements at athmospheric pressure is equal to zero.*

16.5.7 Wheel brake cylinders

The brake cylinders are modeled by the relationship:

$$p = f(\text{liquid volume}) . \quad \textbf{(EQ 224)}$$

The parameters given must describe the sum of the elasticity of the brake cylinder itself as well as of the line between inlet/outlet valve and brake.

The characteristic is parametrized by 3 to 20 points (p, V) as shown in the following diagram. Above and below the given points, the curve is extrapolated linearly.

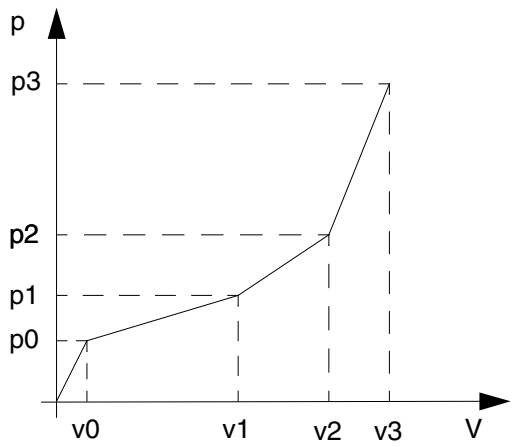


Figure 16.16: Pressure Volume Characteristic

For the following parameters applies: ‘f’ means front , ‘r’ means rear, ‘fl’, ‘fr’ means front left/right and ‘rl’, ‘rr’ means rear left/right.

Cyl_f.pv.mapping
or alternatively:
Cyl_fl.pv.mapping
Cyl_fr.pv.mapping

Cyl_r.pv.mapping
or alternatively:
Cyl_rl.pv.mapping
Cyl_rr.pv.mapping

Vector of pressure values [bar] corresponding to volume values [cm^3]. It is the volume of brake fluid absorbed by the cylinder at a given pressure. It is crucial that the supporting points for the pressure rise strictly monotonic.

Syntax Infofile table mapping with 2 columns
<Pressure> <Volume>

Example Cyl_f.pv.mapping:

0	0.00
5	0.20

Example

```
Cyl_f.pv.mapping:
        0       0.00
        10      0.39
        15      0.56
        20      0.72
        25      0.87
        40      1.28
        60      1.75
        80      2.15
        200     3.84

Cyl_r.pv.mapping:
        0       0.00
        5       0.09
        10      0.16
        15      0.25
        20      0.31
        25      0.38
        40      0.55
        60      0.75
        80      0.91
        200     1.69
```

16.5.8 Low pressure accumulator

The low pressure accumulator accumulates the liquid coming out of the outlet valves before it is pumped back in the brake circuit.

It usually consists of a cylinder with a piston loaded by a spring.

The relation between the absorbed liquid and the resulting pressure is modeled by the following characteristic (the gradients below pMin and above pMax are hard-coded).

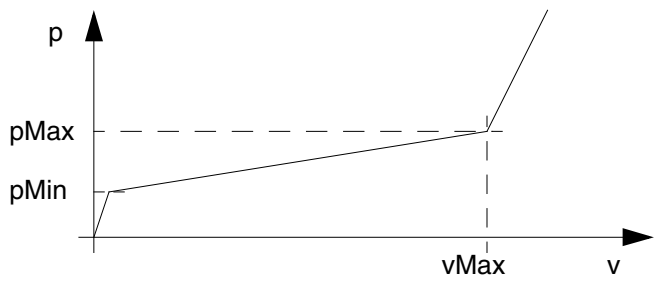


Figure 16.17: Pressures in low pressure accumulator

LPA.vMax

Maximum Volume of pressure accumulator [cm^3].

Example `LPA.vMax = 5.0`

LPA.pMin

Pressure below which the volume is minimal [bar].

Example `LPA.pMin = 1.0`

LPA.pMax

Pressure above which the volume is maximal [bar].

Example `LPA.pMax = 5.0`

16.5.9 Attenuators (damper chambers) and line volumes

The discrete modeling of the hydraulic system makes it necessary to alternate between apertures (flow elements) and volume elements.

Therefore volume elements are positioned between pilot valve, inlet valve and the high pressure side of the pump as well as between low pressure accumulator check valve, suction valve and suction side of the pump. Those elements represent line volumes or attenuator chambers.

They are modeled by proportional characteristics:

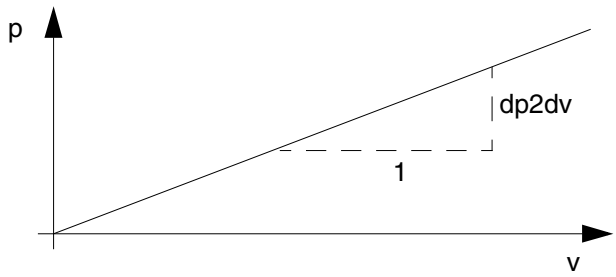


Figure 16.18: Proportional characteristics of attenuators and line volumes

In real brake systems, those line volumes are very small. Nonetheless for numerical reasons, those elements should not be parametrized too stiff, even if the value given doesn't represent the real physical value.

Att.dp2dv

The attenuator is situated at the high pressure side of the pump. For numerical reasons, its value should not be too high $[\mathrm{bar}/\mathrm{cm}^3]$.

$$p = dp2dv \cdot v \qquad \textbf{(EQ 225)}$$

Example `Att.dp2dv = 300.0`

SuppL.dp2dv

The suction line volume is situated at the low pressure side of the hydraulic pump. For numerical reasons, the value given should not be too high [bar/cm^3].

$$p = dp2dv \cdot v \qquad \textbf{(EQ 226)}$$

Example `SuppL.dp2dv = 50.0`

16.5.10 Hydraulic pump (return pump)

The pumps for both circuits are driven by the same motor. The static and dynamic characteristic of the pump can be parametrized.

- Static characteristic:
  - The *flow* is proportional to the rotational velocity. The rotational velocity of the pump decreases with increasing pressure differences. This leads to the following relation:

$$q(\Delta p) = q_{max} - c_{loss} \cdot \Delta p \qquad \text{(EQ 227)}$$

    Alternatively a flow-characteristic can be specified.

$$q(\Delta p) = f(\Delta p, \text{pump activation}) \qquad \text{(EQ 228)}$$

  - *Cavitation on the suction side*. If the relative pressure on the suction side of the pump approaches -1 bar (absolute pressure 0 bar), the flow approaches to zero. This relation is modeled by a exponential saturation function starting at a limit "edge" pressure and a second pressure at which 63% of the full flow are reached:

$$q^* = q\left(1 - e^{\frac{p_{edge} - p}{p_{63} - p_{edge}}}\right) \qquad \text{(EQ 229)}$$

- Dynamic characteristic:

$$\frac{n}{n_{\max}} = \frac{q}{q_{\max}} = 1 - e^{-\frac{t}{\tau}} \qquad \text{(EQ 230)}$$

  When the pump is switched on, the full rotational speed (as well as the full flow) is not reached immediately, but it increases with an exponential saturation function. This function can be parametrized by the time constant τ_{Full} at which 63% of the full flow is reached.

  After switching off the pump, the rotational speed (as well as the flow) decreases exponentially with a time constant τ_{Zero} to be given.

The rotation speed of the hydraulic pump is a relative speed because it is meassured from the voltage ratio of the pump ($\text{current voltage} / \text{maximal voltage}$). It is not possible for the hyrdaulic model to calculate a absolute pump speed.

Usually brake ECUs measure the return voltage generated by the pump to detect pump engine faults. As the pump is a volumetric pump, its rotational speed is approximately proportional to the flow (which is evaluated by the equation or tables above). The model evaluates this voltage as follows:

$$U_{Ret} = \frac{q}{q_{max}} U_{genVmax} \qquad \text{(EQ 231)}$$

with

$U_{genVmax}$	Maximum generated return voltage of the pump (at a pressure difference of 0 bar with no cavitation on the suction side).
q_{max}	Flow at pressure difference 0 bar.

Alternatively a characteristic can be specified.

All Paramaters specified for the hydraulic pump apply to a single brake cricuit (e. g. q_{max} applies per cirquit and is *not* the total flow of both cirquits). But both cirquits are parametized with the same parameter (they are symetrical).

Pump.qMax

Optional instead of `Pump.Flow.mapping`. Maximum delivery of pump. At 0 bar pressure difference $[\mathrm{m}^3/(\mathrm{s}\cdot\mathrm{bar})]$.

Example `Pump.qMax = 5.0`

Pump.cLoss

Optional instead of `Pump.Flow.mapping`. Loss Coefficient of pump. With increase of Δp delivery efficiency will decrease. Small `Pump.cLoss` values are equivalent to a high efficiency characteristic of the pump $[\mathrm{cm}^3/(\mathrm{s}\cdot\mathrm{bar})]$.

Example `Pump.cLoss = 0.01`

Pump.Flow.mapping.Kind

Optional. Determines the type of input for the characteristic `Pump.Flow.mapping`.:

Pump.Flow.mapping.Kind	Description
Standard	Default. One dimensional input characteristic depending on activation signal expected.
PressureSignal	Two dimensional input characteristic depending an activation signal and pressure difference expected.

Example `Pump.Flow.mapping.Kind = PressureSignal`

To obtain reasonable results for a hydraulic pump (flow depends on activation signal/supply voltage and pressure difference) the parameter `Pump.Flow.mapping.Kind` should be set to ‘PressureSignal’.

Pump.Flow.mapping

Optional instead of `Pump.qMax` and `Pump.cLoss`. This characteristic holds lines with the pressure difference at the hydraulic pump [bar] the activation of the pump [0..1] and the resulting delivery flow $[\mathrm{cm}^3/\mathrm{s}]$ of the pump. It is assumed that `Pump.Flow.mapping.Kind` is set to ‘PressureSignal’! .

Syntax Infofile table mapping with 3 columns
<Pressure-Diff.> <Activation-Signal> <Delivery-Flow>

Example Pump.Flow.mapping:

0.0	0.0	0.0
0.0	0.25	6.4
0.0	0.5	6.8
0.0	1.0	7.0
50.0	0.0	0.0
50.0	0.25	5.3
50.0	0.5	6.1
50.0	1.0	6.6
100.0	0.0	0.0
100.0	0.25	4.0
100.0	0.5	5.3
100.0	1.0	6.3
200.0	0.0	0.0
200.0	0.25	1.8
200.0	0.5	3.6
200.0	1.0	5.1

Pump.Full

Time constant τ_{Full} when Pump is accelerating [s]. See (EQ 230).

Example `Pump.Full = 0.1`

Pump.Zero

Time constant τ_{Zero} when pump is deccelerating [s]. See (EQ 230).

Example `Pump.Zero = 0.1`

Pump.pEdge

Only for pressures higher than p_{edge} the Hydraulic pump works. Below p_{edge} no fluid is pumped [bar].

Example `Pump.pEdge = -0.8`

Pump.p63Prcnt

At this pressure the pump operates at 63% of its full delivery capacity [bar].

Example `Pump.p63Prcnt = -0.5`

Pump.genVmax

Optional. Generated voltage of the pump at maximum rotational speed [V], see (EQ 231). Default: 8.0 V.

Example `Pump.genVmax = 8.0`

16.5.11 Valves and Connecting Lines in General

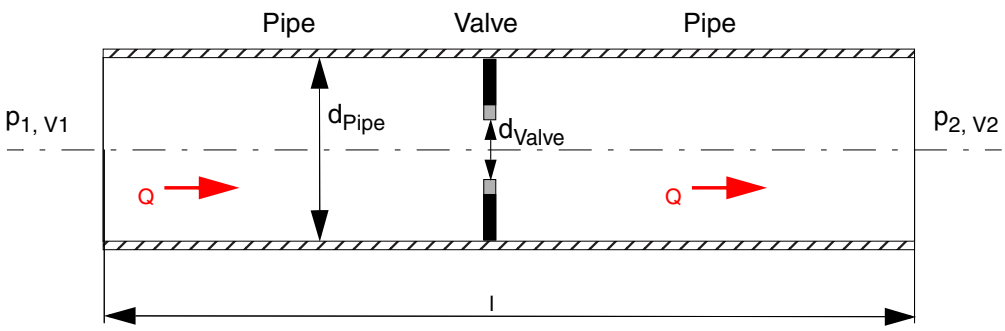


Figure 16.19: Modeling of hydraulic valves and connecting lines

Applied to this model is that a valve with its corresponding pipe is always situated between two hydraulic reservoirs with the state variables pressure and volume. The pressure difference

$$\Delta p = p_1 - p_2 \qquad \text{(EQ 232)}$$

is tried to be equalized by a flow

$$Q = Q_{Pipe} = Q_{Valve} = f(\Delta p, ...) \qquad \text{(EQ 233)}$$

through the pipe and the valve aperture.

The complete pressure difference Δp calculates to:

$$\Delta p = \Delta p_{Valve} + \Delta p_{Pipe} \qquad \text{(EQ 234)}$$

To solve this equation relations for the pressure portions of the flow of Δp_{Valve} and Δp_{Pipe} are needed:

It is assumed that the flow through the pipe is always laminar and through the valve aperture always turbulent.

- Laminar flow through a pipe:

$$Q_{Pipe} = \frac{A d_{Pipe}^2}{32 l \rho \nu} \Delta p = q_{Pipe} \Delta p \qquad \text{(EQ 235)}$$

with

Δp	pressure gradient (= p_{in}- p_{out})
A	cross section area of the pipe
d	diameter pipe
ρ	density
ν	kinematic viscosity
q_{Pipe}	flow coefficient pipe

q_{Pipe} can be calculated out of geometrical data using the following equation:

$$q_{Pipe} = \frac{\pi d_{Pipe}^4}{128 l \rho \nu} \quad \text{(EQ 236)}$$

The value of the parameter q_{Pipe} has to be specified in:

$$[q_{Pipe}] = \frac{cm^3}{s \cdot bar} = 10^{-6} \cdot \frac{m^3}{s \cdot bar} \quad \text{(EQ 237)}$$

- Turbulent flow through a valve:

$$Q_{Valve} = \alpha A \sqrt{\frac{2}{\rho}|\Delta p|} \cdot \operatorname{sgn}\Delta p = q_{Valve}\sqrt{|\Delta p|} \cdot \operatorname{sgn}\Delta p \quad \text{(EQ 238)}$$

with

Δp	pressure gradient (= p_{in}- p_{out})
α	coefficient depending on the geometry of the aperture
A	aperture area
ρ	density
q_{Valve}	flow coefficient valve aperture

q_{Valve} can be calculated out of geometrical data using the following equation:

$$q_{Valve} = \alpha \frac{\pi d_{Valve}^2}{4} \sqrt{\frac{2}{\rho}} \quad \text{(EQ 239)}$$

The value of the parameter q_{Valve} has to be specified in:

$$[q_{Valve}] = \frac{cm^3}{s\sqrt{bar}} = 10^{-6} \cdot \frac{m^3}{s\sqrt{bar}} \quad \text{(EQ 240)}$$

Combining (EQ 233), (EQ 234), (EQ 235) and (EQ 238) leads to:

$$\Delta p(Q) = \left(\frac{Q}{q_{Valve}}\right)^2 + \frac{Q}{q_{Pipe}} \quad \text{(EQ 241)}$$

This relation between pressure difference and flow is valid for all modeled valves. Variations are characterized through q_{Valve} and q_{Pipe}.

16.5.12 Solenoid valves

The solenoid valves are controlled by the power of the magnetic field through the magnetic coils. Basically two different kind of solenoid valves are distinguished:

- **Digital valves** have activation signals that normally either completely open (full activation) or completely close (zero activation) a valve.
- **Proportional valves** also have activation signals other than zero or full activation. An activation between zero and full may lead to intermediate opening states of the valve.

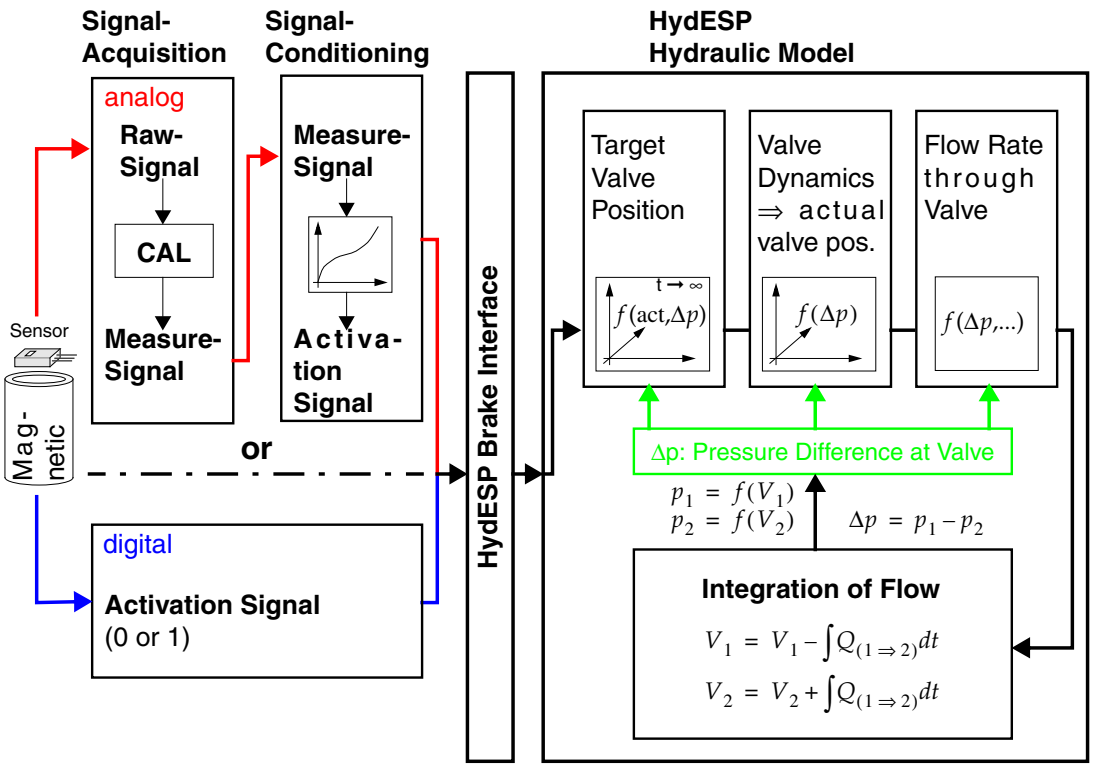


Figure 16.20: From detection to calculation of a hydraulic valve

The interface between the ECU and the simulated hydraulic block is at the magnetic coil. Figure 16.20 shows the process from sensing the valve position up to the calculation of the volume transferred through the valve and the connected pipe.

There are two possibilities for the acquisition of the valve activation signals. For **proportional valves** the acquiring method has to be *analog* in order to capture intermediate valve positions. A *raw signal* captured from the sensor is converted by a calibration function into the measure signal (typically it is the voltage output signal of a hall sensor). After that the measure signal has to be converted into a standardized valve *activation signal*. This is the input signal for each solenoid valve to the *brake interface*.

The same standardized *activation signal* is delivered from the **digital valve** acquisition. It is the digital signal itself which is characterized by the states zero and one. An alternative method for capturing digital valves is to use the analog *signal acquisition*. The *signal conditioning* part is used to define a threshold for valve activity.

After the activation signal of each valve is stored in the *brake interface* structure the calculation process within the brake model can be initiated. The intermediate state (without dynamics, $t \rightarrow \infty$) of a proportional valve is a function of the activation signal and the pressure difference of the valve. This is called the *target valve position*. Because there is no pressure dependency for digital valves this function is a one to one relation of the activation signal.

The time response (so to speak when the target valve position is reached) is calculated by the *valve dynamics* function. The opening time and closing time for the valve can be specified. As well, the opening/closing times are functions of the pressure difference. The result is the standardized *actual valve position*.

The *actual valve position* is input for the flow rate calculation of the valve/pipe system. In this context it is considered as a *relative valve opening*. The relative valve opening α_{rel} [0..1] is defined as:

$$\alpha_{rel} = \frac{q_{valve}}{q_{valve,\,max}} \qquad \textbf{(EQ 242)}$$

$q_{valve,\,max}$ is the maximum flow rate which is reached with a fully opened valve (actual valve position = 1). The calculation of the flow is done by (EQ 241) considering (EQ 242).

With the knowledge of the actual flow through the valve/pipe system the volume change of the two bounding reservoirs can be calculated by integration of flow. This is shown in Figure 16.20 at the lower part of the HydESP section.

The *pressure difference of the valve* is a central part for the calculation. It can be positive or negative. Here, the pressure difference is defined to be positive if the input pressure is higher than the output pressure, with the following definitions of inputs and outputs:

valve	input side	output side
inlet valve	pressure side pump	wheel brake
outlet valve	wheel brake	low pressure accumulator
pilot valve	pressure side pump	master cylinder
suction valve	master cylinder	suction side pump

16.5.13 Proportional Solenoid Valves

All *solenoid valves* can be proportional or digital valves. For the *proportional valves* a transfer mapping characteristic has to be specified to obtain the relative opening of the valve depending on the activation signal and the current pressure difference at the valve.

A relative opening of 1 means that the valve is completely opened, 0 means completely closed. This is important because there are valves that are *closed* when *activated* (activation signal 1 results in a relative opening of 0), or *opened* when *activated* (activation signal 1 results in a relative opening of 1)

Inlet_f.transfer.mapping.Kind
or alternatively:
Inlet_fl.transfer.mapping.Kind
Inlet_fr.transfer.mapping.Kind

Inlet_r.transfer.mapping.Kind
or alternatively:
Inlet_rl.transfer.mapping.Kind
Inlet_rr.transfer.mapping.Kind

Outlet_f.transfer.mapping.Kind
or alternatively:
Outlet_fl.transfer.mapping.Kind
Outlet_fr.transfer.mapping.Kind

Outlet_r.transfer.mapping.Kind
or alternatively:
Outlet_rl.transfer.mapping.Kind
Outlet_rr.transfer.mapping.Kind

PV.transfer.mapping.Kind
SV.transfer.mapping.Kind

Optional. Determines the type of input for the characteristic `Inlet_<f|r>.transfer.mapping.`:

.transfer.mapping.Kind	Description
Standard	Default. One dimensional input characteristic depending on activation signal expected.
PressureSignal	Two dimensional input characteristic depending an activation signal and pressure difference expected.

Inlet_f.transfer.mapping:
or alternatively:
Inlet_fl.transfer.mapping:
Inlet_fr.transfer.mapping:

Inlet_r.transfer.mapping:
or alternatively:
Inlet_rl.transfer.mapping:
Inlet_rr.transfer.mapping:

Outlet_f.transfer.mapping:
or alternatively:
Outlet_fl.transfer.mapping:
Outlet_fr.transfer.mapping:

Outlet_r.transfer.mapping:
or alternatively:
Outlet_rl.transfer.mapping:
Outlet_rr.transfer.mapping:

PV.transfer.mapping:
SV.transfer.mapping:

Optional. Used to calculate the target valve position of the valve. See Figure 16.20.
Syntax depending on `Inlet_<f|r>.transfer.mapping.Kind` (see above):

- Standard:

Characteristic specifying values for relative valve opening [0..1] depending on valve activation signal [0..1]..

Syntax
Infofile table mapping with 2 columns
<Act.-Signal> <Rel. Opening>

Example
Inlet_<f|r>.transfer.mapping:

0.0	1.0
0.5	0.6
1.0	1.0

- PressureSignal:

  Characteristic specifying values for relative valve opening [0..1] depending on pressure difference [bar] and valve activation signal [0..1].

Syntax
Infofile table mapping with 3 columns
<Delta_p> <Act.-Signal> <Rel. Opening>

Example
Inlet_f.transfer.mapping.Kind = PressureSignal
Inlet_f.transfer.mapping:

0.0	0.3	1.0
0.0	0.8	0.0
100	0.3	1.0
100	0.8	0.0

16.5.14 Dynamic Solenoid Valves

All *solenoid valves* no matter if they are proportional or digital can have a different dynamic behavior depending on the current pressure difference at the valve. At higher pressures valves usually open faster and close slower. This behavior is specified with the following parameters:

Inlet_f.deltaT.mapping:
or alternatively:
Inlet_fl.deltaT.mapping:
Inlet_fr.deltaT.mapping:

Inlet_r.deltaT.mapping:
or alternatively:
Inlet_rl.deltaT.mapping:
Inlet_rr.deltaT.mapping:

Outlet_f.deltaT.mapping:
or alternatively:
Outlet_fl.deltaT.mapping:
Outlet_fr.deltaT.mapping:

Outlet_r.deltaT.mapping:
or alternatively:
Outlet_rl.deltaT.mapping:
Outlet_rr.deltaT.mapping:

PV.deltaT.mapping:
SV.deltaT.mapping:

Optional. Used to calculate valve dynamics and the actual valve position. Characteristic specifying values for opening time [s] and closing time [s] of the valve depending on the pressure difference at the valve [bar]. The durations specify the time for open a completely closed valve and vice versa.

Syntax

Infofile table mapping with 3 columns

<Delta_p>	<Opening Time>	<Closing Time>

Example

IInlet_f.deltaT.mapping:

100	0.002	0.01
200	0.001	0.01
0	0.01	0.01

Inlet Valves with check valve

The inlet valve (or supply valve) is situated before the wheel brake. A check valve is positioned parallel to the inlet valve. It is opened if the pressure in the wheel brake becomes higher than on the inlet side. When not driven, inlet valves are opened (flow through valve is possible).

Usually, solenoid valves are digital or proportional 2/2-valves. But there are also systems with valves that have a variable aperture. Those valves have a small and a big aperture, the switching between them is done by a mechanical system depending on the pressure difference:

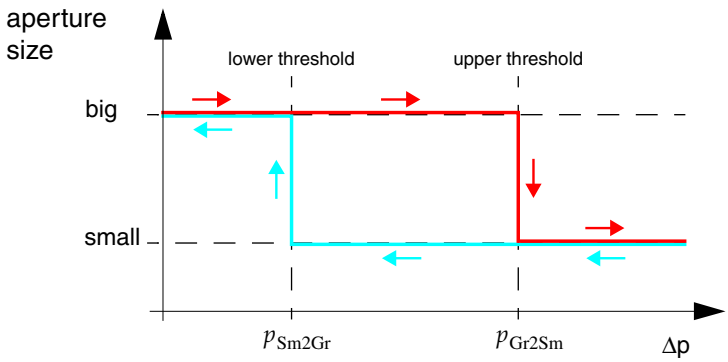


Figure 16.21: Switching between different aperture sizes

pressure difference	switch to
lower than lower threshold	big aperture
higher than upper threshold	small aperture

For the following parameters applies: ‘f’ means front , ‘r’ means rear, ‘fl’, ‘fr’ means front left/right and ‘rl’, ‘rr’ means rear left/right.

For valves with unswitched apertures:

Inlet_f.qOri
or alternatively:
Inlet_fl.qOri
Inlet_fr.qOri

Inlet_r.qOri
or alternatively:
Inlet_rl.qOri
Inlet_rr.qOri

Flow coefficient of valve without switched apertures ('Ori' = Orifice = Aperture) $[\mathrm{cm}^3/(\mathrm{s}\cdot\sqrt{\mathrm{bar}})]$.

Example `Inlet_f.qOri = 4.0`

For valves with switched apertures:

Inlet_f.qOriGr
or alternatively:
Inlet_fl.qOriGr
Inlet_fr.qOriGr

Inlet_r.qOriGr
or alternatively:
Inlet_rl.qOriGr
Inlet_rr.qOriGr

Flow coefficient of valve when switched to big aperture ('Ori' = Orifice = Aperture) $[\mathrm{cm}^3/(\mathrm{s}\cdot\sqrt{\mathrm{bar}})]$.

Example `Inlet_f.qOriGr = 4.0`

Inlet_f.qOriSm
or alternatively:
Inlet_fl.qOriSm
Inlet_fr.qOriSm

Inlet_r.qOriSm
or alternatively:
Inlet_rl.qOriSm
Inlet_rr.qOriSm

Flow coefficient of valve when switched to small aperture ('Ori' = Orifice = Aperture) $[\mathrm{cm}^3/(\mathrm{s}\cdot\sqrt{\mathrm{bar}})]$.

Example `Inlet_f.qOriGr = 2.8`

Inlet_f.pSm2Gr
or alternatively:
Inlet_fl.pSm2Gr
Inlet_fr.pSm2Gr

Inlet_r.pSm2Gr
or alternatively:
Inlet_rl.pSm2Gr
Inlet_rr.pSm2Gr

Limit pressure below which valve switches from “small” to “big” aperture [bar].
See Figure 16.21.

Example `Inlet_f.pSm2Gr = 14.0`

Inlet_f.pGr2Sm
or alternatively:
Inlet_fl.pGr2Sm
Inlet_fr.pGr2Sm

Inlet_r.pGr2Sm
or alternatively:
Inlet_rl.pGr2Sm
Inlet_rr.pGr2Sm

Limit pressure above which valve switches from “big” to “small” aperture [bar].
See Figure 16.21.

Example `Inlet_f.pGr2Sm = 18.0`

For all inlet valves:

Inlet_f.qPipe
or alternatively:
Inlet_fl.qPipe
Inlet_fr.qPipe

Inlet_r.qPipe
or alternatively:
Inlet_rl.qPipe
Inlet_rr.qPipe

Optional. Overall flow coefficient of pipe in front and behind the valve $[\mathrm{cm}^3/(\mathrm{s}\cdot\mathrm{bar})]$. Default: 10^6.

Example `Inlet_f.qPipe = 10000`

Inlet check valves:

InCheckV_f.qOri
or alternatively:
InCheckV_fl.qOri
InCheckV_fr.qOri

InCheckV_r.qOri
or alternatively:
InCheckV_rl.qOri
InCheckV_rr.qOri

Flow coefficient of check valve situated beside inlet valve ('Ori' = Orifice = Aperture) $[\mathrm{cm}^3/(\mathrm{s}\cdot\sqrt{\mathrm{bar}})]$.

Example `InCheckV_f.qOri = 6.0`

InCheckV_f.qPipe
or alternatively:
InCheckV_fl.qPipe
InCheckV_fr.qPipe

InCheckV_r.qPipe
or alternatively:
InCheckV_rr.qPipe
InCheckV_rl.qPipe

Optional. Overall flow coefficient of pipe of check valve $[\mathrm{cm}^3/(\mathrm{s}\cdot\mathrm{bar})]$. Usually this is the same as `Inlet_<f|r>.qPipe`. Default: 10^6.

Example `InCheckV_f.qPipe = 20000`

Outlet Valves

The outlet valves (or: discharge valves) are situated behind the wheel brake. When not driven, the outlet valves are closed.

Outlet_f.qOri
or alternatively:
Outlet_fl.qOri
Outlet_fr.qOri

Outlet_r.qOri
or alternatively:
Outlet_rl.qOri
Outlet_rr.qOri

Flow coefficient of outlet valve ('Ori' = Orifice = Aperture) $[\mathrm{cm}^3/(\mathrm{s}\cdot\sqrt{\mathrm{bar}})]$.

Example `Outlet_f.qOri = 4.0`

Outlet_f.qPipe
or alternatively:
Outlet_fl.qPipe
Outlet_fr.qPipe

Outlet_r.qPipe
or alternatively:
Outlet_rl.qPipe
Outlet_rr.qPipe

Optional. Overall flow coefficient of pipe before and after the outlet valve $[\mathrm{cm}^3/(\mathrm{s}\cdot\mathrm{bar})]$. Default: 10^6.

Example `Outlet_f.qPipe = 100000`

16.5.15 Pilot valve with check valve and pressure limiting valve

The pilot valves are used to separate the brake circuit from the master cylinder. This allows the system autonomously to generate a brake pressure in the circuit. When not driven, the pilot valves are open.

A check valve is positioned parallel to the pilot valve. It opens if the pressure in the master cylinder is higher than the one in the circuit.

A pressure limiting valve is positioned parallel to the pilot valve as well. It opens if the pressure in the brake circuit is higher than the pressure limit. In this case, the flow is calculated as follows:

$$Q_{PLim} = q_{PLimValve} \cdot \sqrt{|\Delta p| - p_{Limit}} \cdot \operatorname{sgn} \Delta p \qquad \textbf{(EQ 243)}$$

The pressure limiting valve is then modeled by the relationship

$$\Delta p(Q_{PLim}) = p_{Open} + \left(\frac{Q}{q_{PLimValve}}\right)^2 + \frac{Q}{q_{PLimPipe}} \qquad \textbf{(EQ 244)}$$

Pure ABS systems don't have pilot valves. For ABS systems use a relatively high value for the flow coefficient. Be careful with too high values and check numerical stability.

PV.qOri

Flow coefficient of pilot valve ('Ori' = Orifice = Aperture) $[\mathrm{cm}^3/(\mathrm{s} \cdot \sqrt{\mathrm{bar}})]$.

Example `PV.qOri = 5.0`

PV.qPipe

Optional. Overall flow coefficient of pipe before and after the pilot valve $[\mathrm{cm}^3/(\mathrm{s} \cdot \mathrm{bar})]$. Default: 10^6.

Example `PV.qPipe = 20000`

PLim.qOri

Flow coefficient of pressure limiting valve ('Ori' = Orifice = Aperture) $[\mathrm{cm}^3/(\mathrm{s} \cdot \sqrt{\mathrm{bar}})]$.

Example `PLim.qOri = 5.0`

PLim.qPipe

Optional. Overall flow coefficient of pipe before and after the pressure limiting valve $[\mathrm{cm}^3/(\mathrm{s} \cdot \mathrm{bar})]$. Usually set to the same value as `PV.qPipe`. Default: 10^6.

Example `PLim.qPipe = 20000`

PLim.pOpen

At a pressure difference superior to PLim.pOpen, the pressure limiting valve opens [bar].

Example `PLim.pOpen = 180.0`

PVcheckV.qOri

Flow coefficient of check valve (‘Ori’ = Orifice = Aperture) $[\mathrm{cm}^3/(\mathrm{s}\cdot\sqrt{\mathrm{bar}})]$.

Example `PVcheckV.qOri = 5.0`

PVcheckV.qPipe

Optional. Overall flow coefficient of pipe before and after the check valve $[\mathrm{cm}^3/(\mathrm{s}\cdot\mathrm{bar})]$. Usually set to the same value as `PV.qPipe`. Default:10^6.

Example `PVcheckV.qPipe = 20000`

16.5.16 Suction valve

In case of an autonomous brake pressure generation, this valve enables the hydraulic pump to take liquid out of the master cylinder. When not driven, they are closed. The suction valve is situated between the master cylinder and the low pressure side of the hydraulic pump.

Pure ABS systems don have suction valves. For ABS systems use a relatively high value for the flow coefficient. Be careful with too high values and check numerical stability.

SV.qOri

Flow coefficient of suction valve (‘Ori’ = Orifice = Aperture) $[\mathrm{cm}^3/(\mathrm{s}\cdot\sqrt{\mathrm{bar}})]$.

Example `SV.qOri = 10.0`

SV.qPipe

Optional. Overall flow coefficient of pipe before and after the suction valve $[\mathrm{cm}^3/(\mathrm{s}\cdot\mathrm{bar})]$. Default: 10^6.

Example `SV.qPipe = 20000`

16.5.17 Check Valve of the Low Pressure Accumulator

This valve is situated between the low pressure accumulator and the low pressure side of the hydraulic pump. The purpose of this valve is to inhibit the flow trough the suction valve to the low pressure accumulator. It opens when the pressure on the low pressure accumulator side is higher than the one on the hydraulic pump side.

LPAcheckV.qOri

Flow coefficient of low pressure accumulator check valve ('Ori' = Orifice = Aperture) $[\mathrm{cm}^3/(\mathrm{s}\cdot\sqrt{\mathrm{bar}})]$.

Example `LPAcheckV.qOri = 6.0`

LPAcheckV.qPipe

Optional. Overall flow coefficient of pipe before and after the low pressure accumulator check valve $[\mathrm{cm}^3/(\mathrm{s}\cdot\mathrm{bar})]$. Default: 10^6.

Example `LPAcheckV.qPipe = 20000`

16.5.18 Temperature of the Brake Fluid

In consequence of a temperature change of the hydraulic fluid the mass density ρ and the kinematic viscosity ν of the fluid change as well. Apparently the equations (EQ 235) and (EQ 238) show the following relations:

$$q_{\mathrm{valve}} \sim \frac{1}{\sqrt{\rho}} \text{ and } q_{\mathrm{pipe}} \sim \frac{1}{\rho\nu}. \qquad \textbf{(EQ 245)}$$

This means that parameter values for q_{valve} and q_{pipe} specified in the hydraulic dataset are only valid for a specific reference temperature T_{Ref} which usually equals the 'normal' operation temperature.

Investigations about the significance of the relations $\rho(T)$ and $\nu(T)$ point out clearly that there is a negligible temperature influence of the mass density and a significant temperature dependence of the kinematic viscosity. This means that a change of temperature is only relevant for the flow coefficients of the hydraulic pipes.

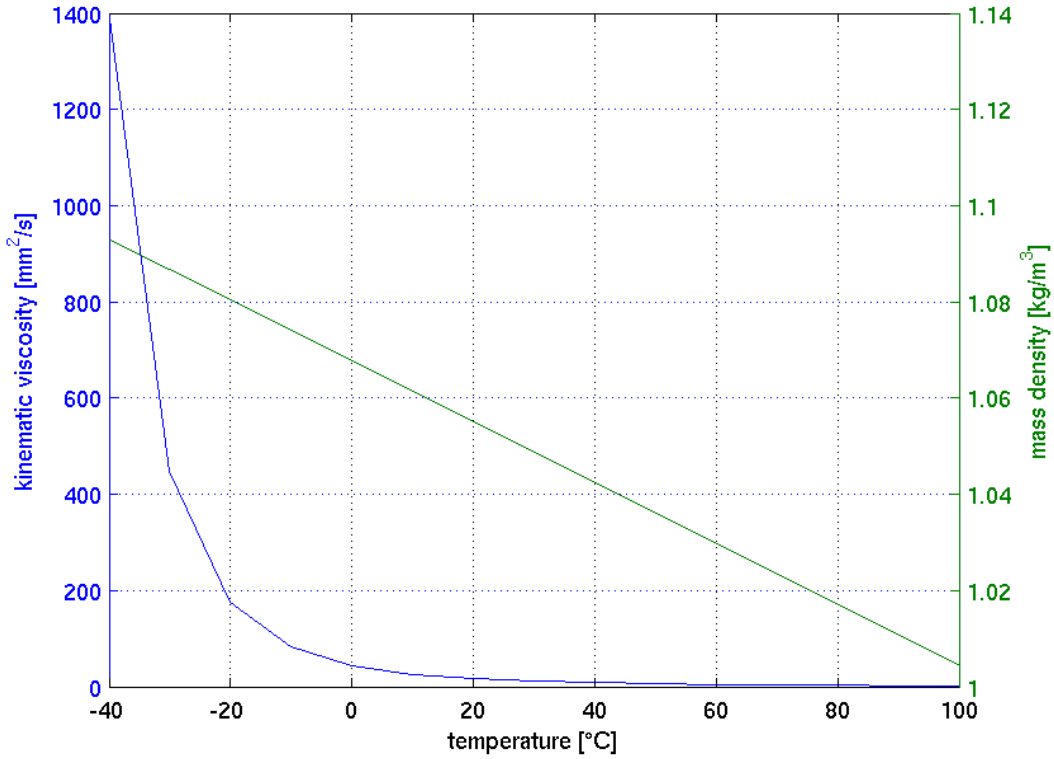


Figure 16.23: Change of kinematic viscosity and mass density of hydraulic fluid as a function of temperature

The flow parameters of the lines are specified by the user for a specific reference temperature. If the brake should operate at a temperature that differs from the reference temperature, the operating temperature can be changed to an alternative value (input from brake interface). The flow characteristics are then adapted automatically by the brake model with the given characteristics for the hydraulic fluid.

In order to parametrize the properties of the brake fluid, kinematic viscosities for two different temperatures have to be given. Those two points are then used by the model to determine the coefficients in its viscosity function.

The hydraulic model uses the following approach to scale the flow coefficients to the specific operating temperature. A factor

$$nf = \frac{\nu_{Ref}}{\nu_T} \qquad \text{(EQ 246)}$$

with

nf	Scaling factor,
v_{Ref}	Kinematic viscosity of brake fluid at reference temperature,
ν_T	Kinematic viscosity of brake fluid at the specified operating temperature of the brake,

is used to recalculate the flow coefficients to

$$q_{\text{pipe, T}} = \text{nf} \cdot q_{\text{pipe}} = \frac{\nu_{\text{Ref}}}{\nu_{\text{T}}} \cdot q_{\text{pipe}} \qquad \textbf{(EQ 247)}$$

To calculate the kinematic viscosities the following approach is used,

$$\log(\log\nu) = c \cdot T + k, \qquad \textbf{(EQ 248)}$$

with the coefficients c, k of the linear equation on the right side:

$$c = \frac{(\log(\log\nu_2)) - (\log(\log\nu_1))}{T_2 - T_1} \qquad k = (\log(\log\nu_1)) - c \cdot T_1 \quad . \qquad \textbf{(EQ 249)}$$

The reference kinematic viscosity is the viscosity according to (EQ 248) for the temperature T_{Ref}:

$$\nu_{\text{Ref}} = 10^{10^{c \cdot T_{\text{Ref}} + k}} \quad . \qquad \textbf{(EQ 250)}$$

Fluid.Ref.Temp

Optional. Reference temperature. Temperature of the hydraulic fluid at which the specified flow coefficients are valid. Dimension: K. Default: no default value, temperature dependency of the brake fluid is switched on by specifying this parameter.
Fluid.Ref.Temp = 293

If the parameter `Fluid.Ref.Temp` is specified the following 4 parameters are mandatory.

The viscosity of the brake liquid has to be indicated for two temperatures:

Fluid.P1.Temp

Temperature T_1 to determine linear equation according to (EQ 248) [K].

Example `Fluid.P1.Temp = 233`

Fluid.P1.nue

Viscosity ν_1 to determine linear equation according to (EQ 248) [mm^2/s].

Example `Fluid.P1.nue = 1150`

Fluid.P2.Temp

Temperature T_2 to determine linear equation according to (EQ 248) [K].

Example `Fluid.P2.Temp = 273`

Fluid.P2.nue

Viscosity ν_2 to determine linear equation according to (EQ 248) [mm^2/s].

Example `Fluid.P2.nue = 40`

16.5.19 Parking Brake

The brake model also includes a simple park brake model:

$$T_{Park\text{Brake}} = T_{ParkTrqMax} \cdot ParkBrake \quad \text{(EQ 251)}$$

Park.Torque_max

This parameter sets the max. park brake torque for all wheels (order: fl, fr, rl, rr).

Example `Park.Torque_max = 0 0 1000 1000`

16.5.20 User Accessible Quantities for Hydraulic Brake System "HydESP"

Please refer to section 'Hydraulic Brake System - HydESP'.

Chapter 17
Tire

17.1 Overview

The quality of a vehicle dynamics simulation is heavily influenced by the capabilities of the tire model. This chapter describes the basic outline of tire models used with CarMaker.

Axis systems of the tire model

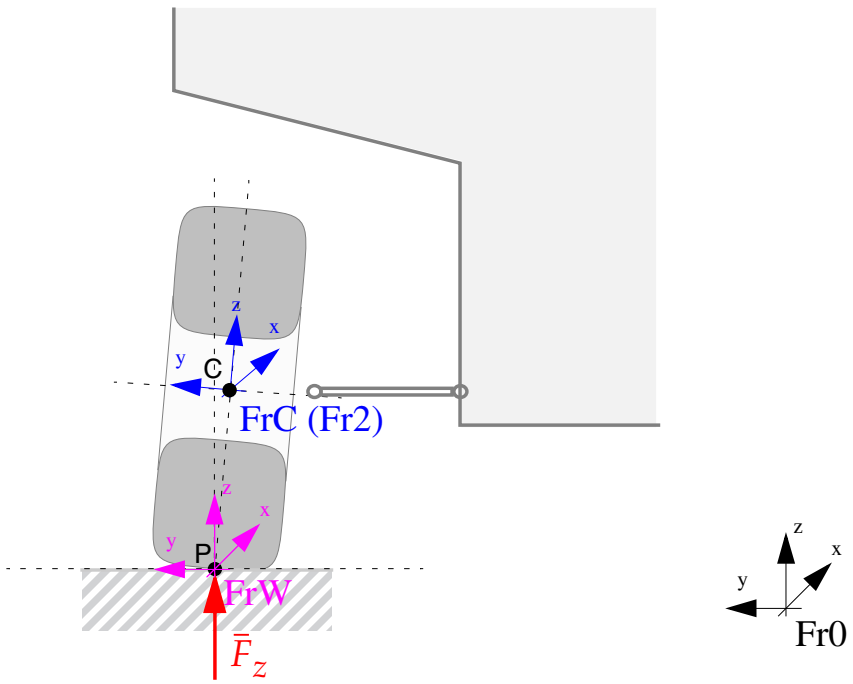


Figure 17.1: Tire axis systems FrW and FrC

There are two points of interest concerning a tire model, the wheel center C and the tire-road contact point P. The axis system Fr2 (corresponds to FrC according to the TYDEX definition), which origins in C, is explained in section 1.2 'CarMaker Axis Systems'. Furthermore there is an axis system FrW which has its origin in the tire-road contact point. FrW is defined according to the TYDEX definition as follows:

- Rules apply to any tire of the vehicle
- (O) is located on the *intersecting line* of the tire vertical plane and the tangent plane of the road. It is defined as the point of the shortest distance $\min(\overline{CP})$ on the *intersecting line*.
- (X) points in forward direction of the *intersecting line* between the tire vertical plane and the tangent plane of the road.
- (Y) points to the left side rectangular to (X) in the tangent plane of the road.
- (Z) points in direction of the normal vector of the tangent plane of the road (mathematically: $(X) \times (Y)$).

CarMaker supports tire models formulated for both the *Contact Point Interface (CPI)* and the *Standard Tire Interface (STI)*.

17.1.1 Tire Model with Contact Point Interface (CPI)

The duty of CPI based tire model is to calculate the tire response forces and torques in the tire-road contact point. So this kind of model does not describe the entire tire with its vertical dynamics, but the contact point between tire and road. Basically the tire forces and torques are functions of:

$$F, T = f(F_z, \alpha, s, \gamma, \mu, \varphi_t ...) \text{ with} \quad \textbf{(EQ 252)}$$

F_z	Normal force in tire contact point
α	Sideslip angle
s	Longitudinal slip
γ	Inclination angle
φ_t	Turn slip
μ	Friction coefficient between tire and road

Interface Structure for CPI based Tire Models

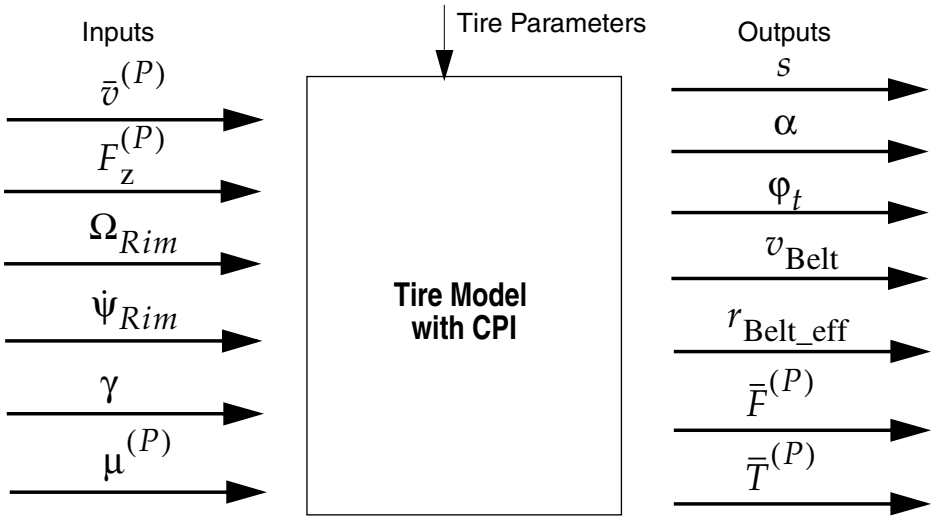


Figure 17.2: Structure of CPI based tire model

The vehicle model provides the tire model with the inputs depicted in Figure 17.2.:

Symbol	Frame	IF-Variable IF.xxx	Description
Inputs			
$\bar{v}^{(P)}$	FrW	P_v0_W[0..2]	Velocity vector in the tire-road contact point
Ω_{Rim}	FrC	Rim_rotv	Rim rotation speed
$\dot{\psi}_{Rim}$	FrW	Rim_turnv	Rim turning speed around vertical axis
$F_z^{(P)}$	FrW	Frc_W[2]	Normal force in tire contact point
γ		InclinAngle	Inclination angle
$\mu^{(P)}$		muRoad	Friction coefficient between tire and road
Outputs			
α		Alpha	Sideslip angle
s		Slp	Longitudinal slip
φ_t		TurnSlp	Turn slip
v_{Belt}		vBelt	Velocity of the tire belt in the tire-road contact point
r_{Belt_eff}		rBelt_eff	Effective rolling radius

Symbol	Frame	IF-Variable IF.xxx	Description
$\bar{F}^{(P)}$	FrW	Frc_W[0..2]	Tire forces in the tire-road contact point
$\bar{T}^{(P)}$	FrW	Trq_W[0..2]	Tire torques in the tire-road contact point

Tire Load (normal force)

The tire load is calculated by the vehicle model and is direct input to the tire model. This section explains how the tire normal force is calculated.

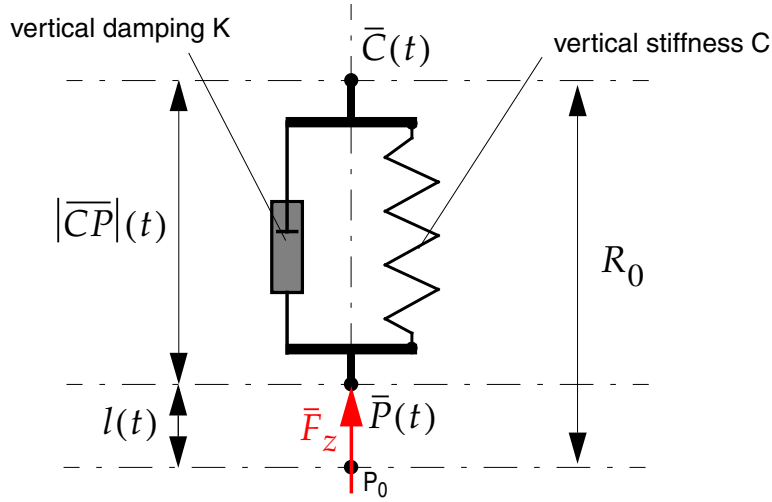


Figure 17.3: Calculation of tire normal load

Tire Model Computations

The job of the tire model is to deduce the parameters of (EQ 252) out of the inputs of the tire interface. This means that sideslip angle and longitudinal slip have to be calculated by the tire model. This has to be done by a relation:

$$\alpha, s = f(\bar{v}^{(P)}, v_{\text{Belt}}) \text{ with } v_{\text{Belt}} = f(\Omega_{Rim}, r_{\text{Belt_eff}}) \quad \textbf{(EQ 253)}$$

The tire model offered by CarMaker uses for the longitudinal slip computation the TYDEX definition:

$$s = \frac{\Omega_{Rim} \cdot r_{\text{Belt_eff}} - v_x^{(P)}}{v_x^{(P)}} \quad \textbf{(EQ 254)}$$

For moving in reverse the computation is modified to:

$$s = \frac{\Omega_{Rim} \cdot r_{\text{Belt_eff}} - v_x^{(P)}}{\left|v_x^{(P)}\right|} \quad \textbf{(EQ 255)}$$

With this definition the longitudinal force Fx and the longitudinal slip have the same sign. When the sign is positive, the vehicle is being forced forward and when it's negative the vehicle is being forced backwards.

For the slip angle computation following definition is used:

$$\alpha = \text{atan}\frac{v_y^{(P)}}{\left|v_x^{(P)}\right|} \quad \textbf{(EQ 256)}$$

The turn slip is calculated as follows:

$$\varphi_t = \frac{\dot{\psi}_{Rim}}{\left|v_x^{(P)}\right|} \quad \text{(EQ 257)}$$

17.1.2 Tire Model with Standard Tire Interface (STI)

A tire model using the Standard Tire Interface (defined by the TYDEX group) has four main duties while simulation phase:

- Obtain the tire-road contact point by calling the road model with the local road normal and friction coefficient.
- Calculate the slip, slip angle, inclination angle and the tire normal force in the tire-road contact point.
- Calculate the tire forces and torques in the tire-road contact point.
- Transformation of the tire forces and torques to the wheel carrier.

So unlike tire models with Contact Point Interface, tire models with Standard Tire Interface describe the entire tire behavior including vertical and horizontal tire characteristics.

The following table and figure show the inputs and outputs of such a tire model. Please note that the STI standard defines only torque and forces to the wheel center (marked with *) as mandatory outputs, but the CarMaker simulation needs some more information about the tire e.g. for IPGDriver, animation with IPGMovie and equilibrium state calculation.

Symbol	Frame	IF-Variable IF.xxx	Description
Inputs			
$\bar{t}^{(C)}$	Fr0	WC_t_0[0..2]	Global coordinates of the wheel center w.r.t. the earth-fixed axis system Fr0
$\bar{v}^{(C)}$	FrC	WC_vel_2[0..2]	Global wheel carrier velocity in the wheel center calculated in the earth-fixed axis system Fr0, expressed in Frame C
$\bar{\Omega}^{(C)}$	FrC	WC_omega_2[0..2]	Global wheel carrier rotation speed in the wheel center calculated in the earth-fixed axis system Fr0, expressed in the Frame C
θ_{Rim}	FrC	Rim_rot	Rotation angle of the rim
Ω_{Rim}	FrC	Rim_rotv	Rotation speed of the rim
$T_{FrC \rightarrow Fr0}$		WC_Tr2Fr0[0..2][0..2]	Transformation matrix from wheel carrier system FrC (Fr2) to earth-fixed axis system Fr0, expressed in C-standard (row-wise)
Outputs			
$\bar{F}^{(C)}$ *	FrC	Frc_C[0..2]	Applied tire forces to the wheel center
$\bar{T}^{(C)}$ *	FrC	Trq_C[0..2]	Applied tire torques to the wheel center
$\bar{F}^{(P)}$	FrW	Frc_W[0..2]	Tire forces in the tire-road contact point
$\bar{T}^{(P)}$	FrW	Trq_W[0..2]	Tire torques in the tire-road contact point
s		Slp	Longitudinal slip
α		Alpha	Sideslip angle
φ_t		TurnSlp	Turn slip
γ		InclinAngle	Inclination angle
$\bar{v}^{(P)}$	FrW	P_v0_W[0..2]	Velocity vector in the tire-road contact point

Symbol	Frame	IF-Variable IF.xxx	Description
v_{Belt}		vBelt	Velocity of the tire belt in the tire-road contact point
$r_{\mathrm{Belt_eff}}$		rBelt_eff	Effective rolling radius
r		Radius	Loaded radius
$\mu^{(P)}$		muRoad	Friction coefficient between tire and road
$\bar{t}^{(C)}$	Fr0	P_0	Global coordinates of the contact point w.r.t. the earth-fixed axis system Fr0
$dist$		sRoad	Road coordinate

Interface Structure of STI based Tire Model

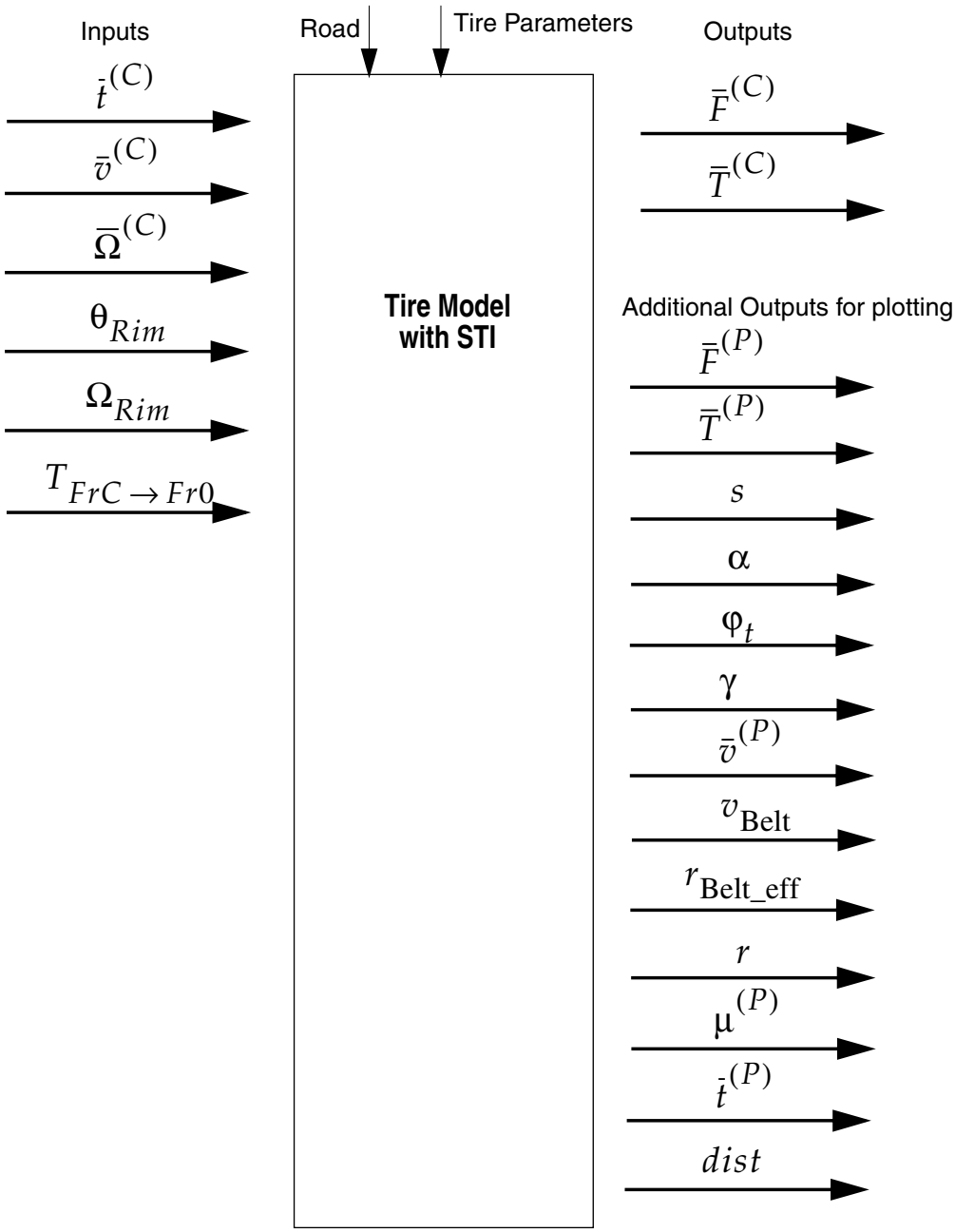


Figure 17.4: Structure of STI based tire model

17.1.3 Overview of Tire Model Calculation in CarMaker

For the calculation of tire forces and torques in the wheel carrier, CarMaker calls the function *Tire_Calc ()*, defined in the header file *Tire.h*. Figure 17.6 gives an overview of this function:

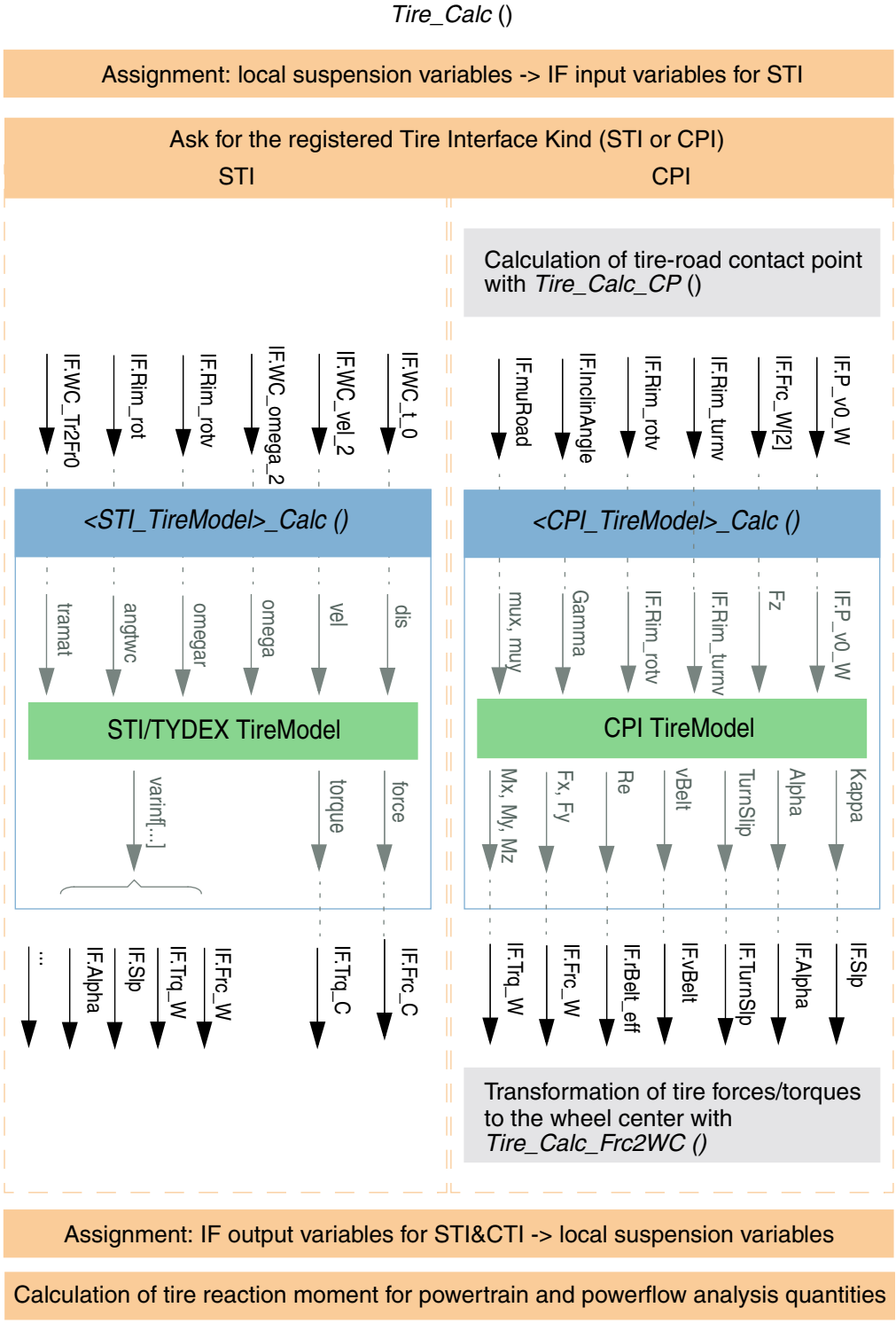


Figure 17.6: Overview of the *Tire_Calc ()* function

17.2 General Tire Parameters

The parameters explained here apply to (various) tire models. All tire parameters are stored in a separate tire data file.

General **FileIdent = CarMaker-Tire-*KindStr Version***

The first parameter has to be the FileIdent parameters which specifies the type and (IPG internal) version of tire model to be used. See section 2.2.1 'File identification' for details.

ModelName	KindStr	IF	Version	Description
IPGTire	IPGTIRE	STI	2	Tire model developed by IPG using TYDEX format for the measured tire characteristic.
RealTime Tire	RTTire	CPI	2,3	RealTime version of IPGTire
Magic Formula	MF52	CPI	3	Pacejka's Magic Formula 5.2
Magic Formula	MF61	CPI	1	Pacejka's Magic Formula 6.1
Tame Tire	TameTire	CPI	1	Thermo mechanical tire model by Michelin
MF-Tyre/MF-Swift 6.2	TNO_MF62	STI	1	MF-Tire/MF-Swift 6.2 tire model by TNO (only windows)
MF-Tyre/MF-Swift 7.2	TASS_MF	STI	1	MF-Tire/MF-Swift 7.2 tire model by TASS International
FTire	FTire	STI	1	FTire (Flexible Structure Tire Model) by cosin

Example `CarMaker-Tire-RTTire 3`

FileCreator = ***Comment***

Optional. String containing information about the file creator.

Description: ***Text***

Description contains formal information about tire details, e. g. tire dimensions, inflation pressure, conditions of measurement, etc.

Carrier.mass = ***Mass***
Carrier.I = ***InertiaTensor***

Optional. Part in wheel carrier mass.
The support for this parameters depends on the vehicle model.
The place of this body is taken from the vehicle parameter set.

Wheel.mass = ***Mass***
Wheel.I = ***InertiaTensor***

Optional. Part in wheel mass.
The support for this parameters depends on the vehicle model.
The place of this body is taken from the vehicle parameter set.

Visualization

AspectRatio

Optional. Represents ratio between rim radius and witdth [-].
To use either the aspect ratio or directly the rim radius parameter.

RimRadius

Optional. Nominal radius of rim used for the animation tool IPGMovie [m].
To use either the aspect ratio or directly the rim radius parameter.

NomWidth

Optional. Nomimal width of the tire for the animation tool IPGMovie [m].

FLoadMax

Optional. Maximum animated load force of tire. Used to scale the tire force vectors in the animation tool IPGMovie [N].

17.2.1 Contact Point Modification

Using tire models with the Contact Point Interface (CPI) it's possible to modify for each tire independently the tire-road contact point position and the road characteristics of this point (normal vector, friction). On this way for example road roughness or wet ruts can be realized. Furthermore this could be used for realization of a poster testrig.

TireCPMod.Kind =*KindStr Version*

Specifies the model to be used for the tire contact point modification.

17.3 Tire Model "IPGTire"

The *IPGTire* model is a map-based tire model developed by IPG, which uses the STI tire interface. The tire characteristic is described by measurement data using the TYDEX tire format.

17.3.1 Parameters

Following additional parameters are required in the tire Infofile:

ITFName

File name with relative path to the directory of the tire Infofile. Two different file types are supported:

- <name>*.tdx*: tire parameters file with testbed measurements using the TYDEX format
- <name>*.tdx.it*: tire parameters file with coefficients (as binary data) as result of converting the original **.tdx* file using the *ipgtirefit* tool

Parameters in the *.tdx file

Most used parameters in the tire parameters file **.tdx* are described as follows:

***MODELPARAMETERS*

Key	Description	Unit	Default value
OVALLDIA	Tire diameter	m	0.58
KROLRAD	Tire rolling radius	m	0.307
NOMWIDTH	Tire width	m	0.2
ASPRATIO	Tire thickness/width ratio		0.6
ITFYVMY	Fy: friction coefficient		0.9
ITFXAVMY	Fx: friction coeff. (acceleration)		0.9
ITFXDVMY	Fy: friction coeff. (deceleration)		0.9
ITMZVMY	Mz: friction coefficient		-0.1
ITVS	Vertical tire stiffness	N/m	200000
ITVD	Vertical tire damping	Ns/m	1000
ITRLLO	Relaxation length longitudinal	m	0.05
ITRLLA	Relaxation length lateral	m	0.5
ITSSCLO	Longitudinal slip coefficient for vx=0 (stand still)		0.01
ITSSCLA	Lateral slip angle coefficient for vx=0 (stand still)		0.2
ITRORETL	Roll resistance torque coefficient		0
ITTIROMY	Friction coefficient in tire-road contact point		1.0
ITLFYTZ	Length for additional align torque Mz=l*Fy	m	0

Key	Description	Unit	Default value
ITLDCL	Fx spline corrector factor for load		0.25
ITSPCL	Fx spline corrector factor for slip		1.0
ITIACL	Fx spline corrector factor for inclination angle		0.75
ITLDCS	Fy spline corrector factor for load		0.25
ITSPCS	Fy spline corrector factor for slip		1.0
ITIACS	Fy spline corrector factor for inclination angle		0.75
ITLDCTZ	Mz spline corrector factor for load		0.25
ITSPCTZ	Mz spline corrector factor for slip		1.0
ITIACTZ	Mz spline corrector factor for inclination angle		0.75
ITLDCTX	Mx spline corrector factor for load		0.25
ITSPCTX	Mx spline corrector factor for slip		1.0
ITIACTX	Mx spline corrector factor for inclination angle		0.75

***MEASURCHANNELS*

Key	Description	Unit
*<xFrame>: can be empty, **C**, **H** or **W** for the different TYDEX tire frames*		
MEASNUMB	Measurement point number	
LONGSLIP	Longitudinal slip	%
SLIPANGL	Side slip angle	deg
INCLANGL	Inclination angle	deg
ADDPARAM	Additional parameter (to use instead of inclination angle)	
FZ*<xFrame>*	Vertical force	N
FX*<xFrame>*	Longitudinal force	N
FY*<xFrame>*	Lateral force	N
MZ*<xFrame>*	Aligning torque	Nm
MX*<xFrame>*	Overturning torque	Nm

***MEASURDATA*

- Typically there are two measurement data fields:
  - depending on longitudinal slip for FX
  - depending on side slip angle for FY, MX, MZ
- A measurement data can use two (3-D) or three (4-D) input variables like:
  - FX = f(FZ, LONGSLIP) or FX = f(FZ, LONGSLIP, INCLANGL/ADDPARAM)
  - FY = f(FZ, SLIPANGL) or FY = f(FZ, SLIPANGL, INCLANGL/ADDPARAM)
  - MX = f(FZ, SLIPANGL) or MX = f(FZ, SLIPANGL, INCLANGL/ADDPARAM)
  - MZ = f(FZ, SLIPANGL) or MZ = f(FZ, SLIPANGL, INCLANGL/ADDPARAM)
  - Mixed use is possible: one data field with 3 and other with 4 dimensions.
- The model requires at least one measurement channel: preferred the side force FY.

- If only one force channel is available, the other is generated automatically.
- The model supports channels using equidistant vertical force points, keeping constant level or randomly selected input values (point cloud).

17.3.2 Approximation of the tire measurement data

To support randomly selected input values (point cloud) for a measurement point, the model needs at first an approximation using an exponential spline to determinate afterwards the coefficients for a cubic spline. Thus the model is more flexible for the input data.
Below a comparison between a measurement data and the IPGTire model simulation is illustrated.

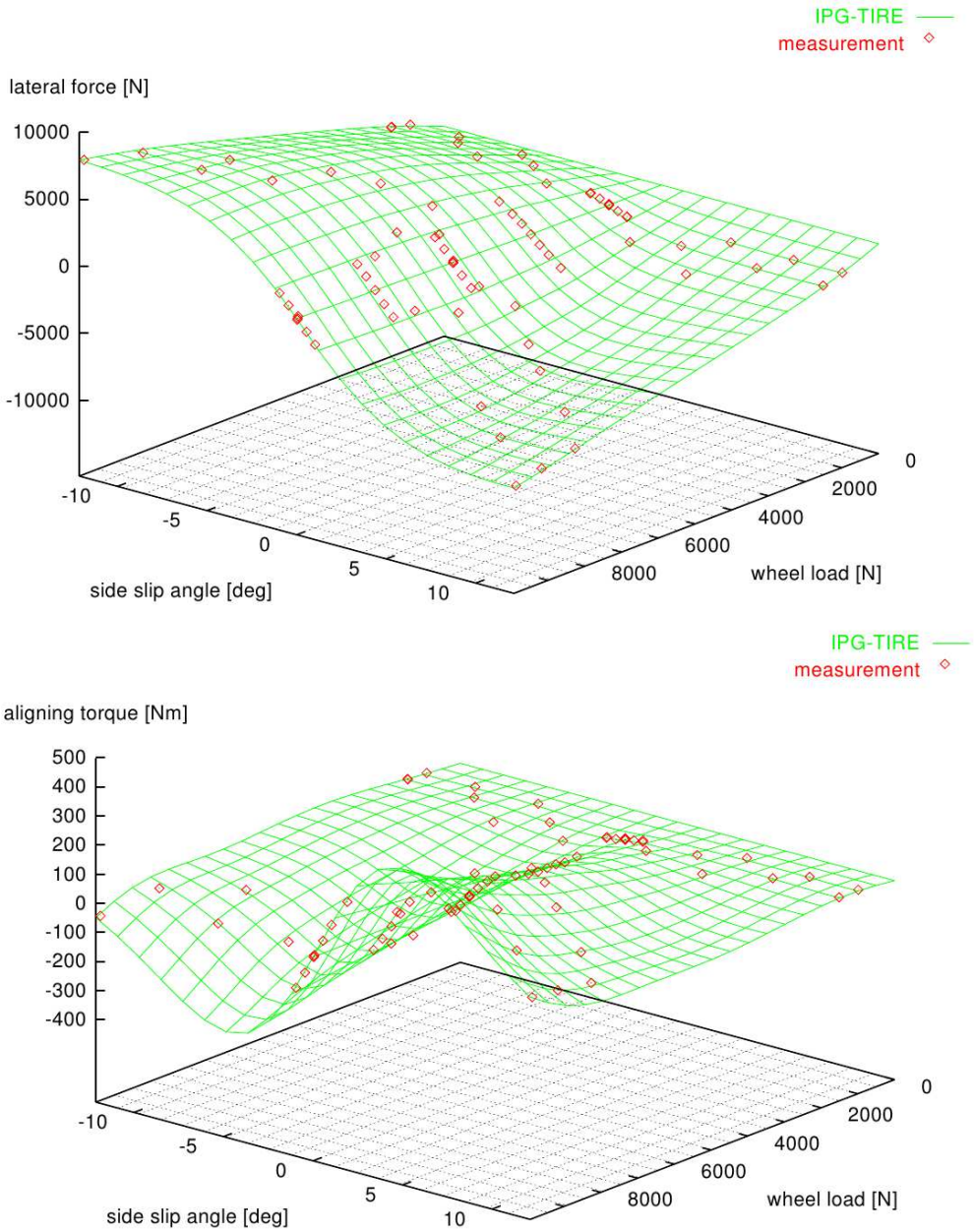


Figure 17.7: Comparison between measurement and IPGTire

17.4 Tire Model "RealTime Tire"

Tire Model *RealTime Tire* is a modified version of IPGTire. This version is optimized for computation speed and designed for using in realtime systems.

Model Ident

muRoad

Friction conditions during tire measurement [-].

BinFName

File name with relative path to directory of the tire infofile. Tire force and torque mappings may be stored in a binary file located in this directory.

Kinematics and Load Force

KinRollRadius

Kinematic tire radius (also known as static tire radius) [m].

NomRadius

Nominal tire radius (also known as unloaded radius) for normal force calculation.
Used also for the animation tool IPGMovie [m].

Radial.Stiffness

Radial/vertical stiffness of tire [N/m].

Radial.Damping

Radial/vertical damping coefficient of tire [N/m/s].

Dynamics

LongFrc.Length
SideFrc.Length
AlignTrq.Length

Specifies the relaxation length to determine the transient response of the longitudinal force Fx, lateral force Fy and aligning torque Tz. The relaxation length ϑ_K is the distance needed to build-up the tire forces and torques. The transient behavior is described using a low pass filter with following time constant: $T = \vartheta_K / |V_x|$.
Unit: [m]. Default: 0.05, 0.1, 0.05.

Stand Still

StandStill.vMax

Optional. Boundary for switching between stand still model and normal computation [m/s]. Default: 0.25.

StandStill.cLong = ***Stiffness***
StandStill.cSide = ***Stiffness***

Optional. Stiffness for stand still model [-]. Default: 0.01, 0.2.

Additional Internal Parameters

LongFrc.Stiffness = ***Factor***
SideFrc.Stiffness = ***Factor***
AlignTrq.Stiffness = ***Factor***

Optional. Scaling factors[-]. Default: 1.0, 1.0, 1.0.

Side= ***side***

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. Possible values are "left" and "right".

Inclination Angle

InclinAngle2OffsetFy

Optional. Inclination angle influence to vertical side force offset. A possible value could be -30. Unit: [N/deg]. Default: 0.0.

InclinAngle2OffsetMz

Optional. Inclination angle influence to vertical aligning torque offset. A possible value could be -0.9. Unit: [Nm/deg]. Default: 0.0.

InclinAngle2Alpha

Optional. Inclination angle influence to sideslip angle. The current inclination angle value multiplied by this factor is added to the sideslip angle value [-]. Effects a horizontal shifting of the side force and aligning torque. A possible value could be 0.03. Unit: []. Default: 0.0.

Turn Slip/ Parking Torque

TurnSlip.MzMax

Optional. Specifies the maximum parking torque for $\varphi_t \rightarrow \infty$.
A possible value could be 200. Unit: [Nm]. Default: 0.0.

TurnSlip.cMz

Optional. Specifies the moment stiffness against turn slip $\partial M_z / \partial \varphi_t |_{\varphi_t = 0}$.
A possible value could be 250. Unit: [Nm^2]. Default: 0.0.

TurnSlip.Alpha2Mz

Optional. Specifies the decline coefficient of the pure turn slip aligning moment depending on sideslip angle. A possible value could be 15. Unit: []. Default: 0.0.

The total aligning moment function is defined as follows:

$$M_z = M_z(\varphi_t)_{\alpha = 0} \cdot \cos(\mathrm{atan}(c_{Alpha2Mz} \cdot \alpha)) + M_z(s, \alpha) \cdot fac(\varphi_t) \quad \textbf{(EQ 258)}$$

$fac(\varphi_t)$ is a factor function to reduce the aligning torque due to sideslip and slip $M_z(s, \alpha)$ linearly with increasing turn slip.

17.4.1 Rolling Resistance

RollResist.Kind = ***KindStr***

The following roll resistance models are known: Torque of force influence based on Load or velocity. There ***KindStr*** are:

KindStr	Description
TrqLoad	A torque along wheel spin axis, against rotation based on load force.
TrqVelocity	A torque along wheel spin axis, against rotation based on translation velocity.
TrqLoadVel	A torque along wheel spin axis, against rotation based on load force, pressure and velocity according SAE J2452.
FLoad	A force along FrW x axis, against translation based on load force.
FVelocity	A force along FrW x axis, against translation based on translation velocity.

To disable roll resistance write **none** or an empty string.

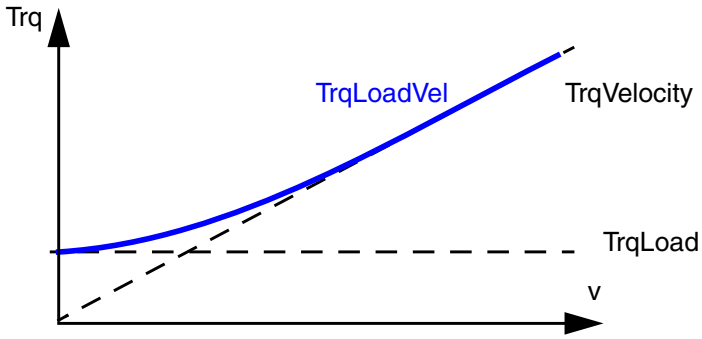


Figure 17.8: Rolling resistance torque depending on velocity

RollResist.Factor

Kind dependent ***Factor***. Input is scaled by ***Factor*** to get the roll resistance output.
Model *TrqLoad* factor is dimensionless [-].
Model *TrqVelocity* factor [Ns/rad]
This parameter is not required by the model *TrqLoadVel*.

Additional Parameters for the rolling resistance model 'TrqLoadVel'

The rolling resistance force according SAE J2452 is defined as follows:

$$F_{RR} = P^{\alpha} F_z^{\beta} \cdot (a + b \cdot v_{Belt} + c \cdot v_{Belt}^{2}) \quad \text{(EQ 259)}$$

with tire parameters $P, \alpha, \beta, a, b, c$ described below. The parameter P is the tire inflation pressure in kPa, F_z is the applied load for vehicle weight in N and v_{Belt} is the tire belt speed in km/h.

The rolling resistance torque along wheel spin axis is calculated as follows:

$$Trq_{RR} = -|F_{RR}| \cdot r \cdot \mathrm{sgn}(v_{Belt}) \quad \text{(EQ 260)}$$

RollResist.TrqLoadVel.Press = ***value***

Tire inflation pressure parameter P in (EQ 259). Unit: [kPa]. Default: 220.

RollResist.TrqLoadVel.alpha = ***value***
RollResist.TrqLoadVel.beta = ***value***

Exponent parameters α, β in (EQ 259). Default: alpha = -0.5; beta = 1.04.

RollResist.TrqLoadVel.a = ***value***
RollResist.TrqLoadVel.b = ***value***
RollResist.TrqLoadVel.c = ***value***

Parameters a, b, c in (EQ 259). Default: a = 1.0e-1; b = 2.0e-4; c = 4.0e-7.

17.4.2 Importing Tire Measurements

The supported input file format is the Tydex format. The Tydex file can be imported by

```
tireutil TydexFileName.tdx
```

The following Tydex model parameters are supported:

ITVS =	***RadialTireStiffness_N/m***
ITVD =	***RadialDamping_N/m/s***
ITRLLO =	***RelaxationLengthLong_m***
ITRLLA =	***RelaxationLengthLateral_m***
ITSSCLO =	***StandStillCoeffLong***
ITSSCLA =	***StandStillCoeffLateral***
ITRORETL =	***RollResistFactor_Trq/Load***

17.5 Tire Model "Magic Formula"

17.5.1 The basics of Magic Formula

When possible “original” Pacejka’s variable and parameter names are used. For background information, you can use publications on Pacjeka tire modeling, magic formula or in Hans B. Pacejka book “Tyre and Vehicle Dynamics” published in november 2002.

The general form of the formula reads:

$$y(x) = D\sin[C\operatorname{atan}\{Bx - E(Bx - \operatorname{atan}Bx)\}] \quad \text{(EQ 261)}$$

with

$Y(X) = y(x) + S_V$

$x = X + S_H$

where:

$Y(X)$: could be either F_x or F_y or possibly M_z (in this case the sine function is replaced by a cosine function.

The variables represents as follows:

B : stiffness factor

C : shape factor

D : peak value

E : curvature factor

S_H : horizontal shift

S_V : vertical shift

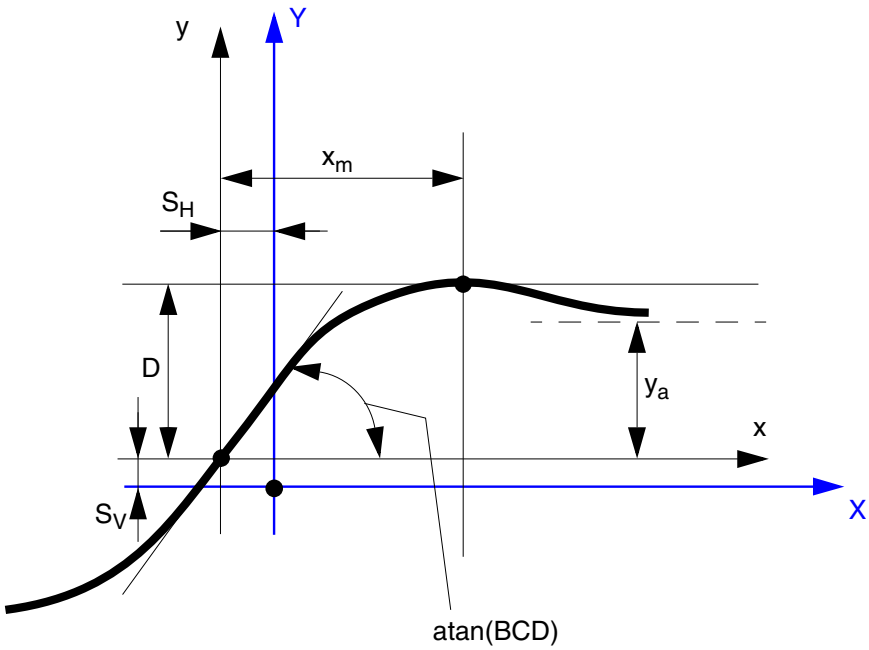


Figure 17.9: Curve produced by the original sine version of the Magic Formula

The Magic Formula $y(x)$ typically produces a curve that passes through the origin $x = y = 0$, reaches a maximum and tends to a horizontal asymptote. To allow the curve to have an offset with respect to origin, two parameters for shifting (S_H and S_V)have been introduced

The MagicFormula model equation contains non-dimensional parameters p ,q ,r and s and additional scaling factors λ. These parameters are used to parametrize a special tire behavior.

The effect of having a tire with different nominal load can be approximated by using a scaling factor:

$$F_{z0}^{'} = \lambda_{Fz0} F_{z0} \quad \text{(EQ 262)}$$

Further, we introduce the normalized change in vertical load

$$df_z = \frac{F_z - F_{z0}^{'}}{F_{z0}^{'}} \quad \text{(EQ 263)}$$

and similarly the normalized change in inflation pressure:

$$dp_i = \frac{P - P_0}{P_0} \quad \text{(EQ 264)}$$

17.5.2 Tire Description

General

AdamsPropertyFile = *PropertyFile*

Optional. Link to an ADAMS® tire property file. Its path is relative to directory of the tire info-file.
Supported Formats: MF_05

Caution.

All parameters listed below can be used to redefine a tire property. This means parameters from *AdamsPropertyFile* have lower priority or will be overwritten by parameters given in the tire data set directly. This feature is very useful if you are tuning a measured tire data set.

Use Mode

Model.USE_MODE

Specifies the forces and torques evaluation mode.

USE_MODE	State	Slip conditions	Calculated forces/ torques
1	Steady state	Pure longitudinal slip	Fx, My
2	Steady state	Pure lateral slip	Fy, Mx, Mz
3	Steady state	Pure longitudinal and lateral (not combined) slip	Fx, Fy, Mx, My, Mz
4	Steady state	Combined slip	Fx, Fy, Mx, My, Mz
5	Steady state	Combined slip + Turn slip	Fx, Fy, Mx, My, Mz
+10	Transient		

Recommended modes for CarMaker are 13,14 or 15.

Model.FITTYP

Selection of the equations. A property file for the version 5.2 must have FITTYP=6, a property file for the version 6.1 must have FITTYP=61.

Valid Range

Range.Consider = *bool*

Optional. Specifies if the following minimum and maximum valid range values are considered. Default: 1.

Range.KPUMIN
Range.KPUMAX

Valid range for the longitudinal slip. Unit: [-]. Longitudinal slip values beside this range are limited to the minimal or maximal value.

Range.FZMAX

Valid range for the normal load [0...FZMAX]. Unit: [N].

Units.ANGLE

Defines the angle unit for the angles range. Possible units are radian (*rad, radian, radians)* or degree (*deg, degree, degrees).* Default: *rad.*

Range.ALPMIN
Range.ALPMAX

Valid range for the sideslip angle. Unit: [Units.ANGLE]. Sideslip angle values beside this range are limited to the minimal or maximal value.

Range.CAMMIN
Range.CAMMAX

Valid range for the inclination angle. Unit: [Units.ANGLE]. Inclination angle values beside this range are limited to the minimal or maximal value.

Range.PRESMIN
Range.PRESMAX

Valid range for the inflation pressure. Unit: [Pa]. Inflation pressure values beside this range are limited to the minimal or maximal value.

Dimensions and Visualization

Dim.UNLOADED_RADIUS

Non-rolling free tire radius R_0 also used for the animation tool IPGMovie [m].

Dim.WIDTH

Tire width also used for the animation tool IPGMovie [m].

Dim.RIM_RADIUS

Rim radius also used for the animation tool IPGMovie [m].

Dim.RIM_WIDTH

Rim width also used for the animation tool IPGMovie [m].

Dim.ASPECT_RATIO

Aspect Ratio = tire height / tire width [-].

FLoadMax

Optional. Maximum animated load force of tire. Used to scale the tire force vectors in the animation tool IPGMovie [N].

Vertical.VERTICAL_STIFFNESS

Radial/vertical stiffness C_{z0} of tire [N/m].

Vertical.VERTICAL_DAMPING

Radial/vertical damping coefficient of tire [N/m/s].

Operating Conditions

OperCond.NOMPRES

Nominal tire inflation pressure P_0 used in equations [Pa].

OperCond.INFLPRES

Current tire inflation pressure P [Pa].

Stand Still

StandStill.cLong = ***Stiffness***
StandStill.cSide = ***Stiffness***

Optional. Stiffness of the stand still model [-]. Default: 0.01, 0.2.

StandStill.vMax

Optional. Boundary for switching between stand still model and normal computation [m/s]. Default: 0.25 m/s.

Additional Internal Parameters

Side= ***side***

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.

Example `Side = left`

Friction Coefficient

muRoad

Friction conditions during tire measurement. Unit: [-]. Default: 1.0.

17.5.3 Scale factors

Scale

Scale.LFZO

Optional. Scale factor of nominal load [-]. Default: 1.
Please note: The parameter name ends with the letter O (and not with a zero).

Scale.LCX

Optional. Scale factor of F_x shape factor [-]. Default: 1.

Scale.LMUX

Optional. Scale factor of F_x peak friction coefficient [-]. Default: 1.

Scale.LEX

Optional. Scale factor of F_x curvature factor [-]. Default: 1.

Scale.LKX

Optional. Scale factor of F_x slip stiffness [-]. Default: 1.

Scale.LHX

Optional. Scale factor of F_x horizontal shift [-]. Default: 1.

Scale.LVX

Optional. Scale factor of F_x vertical shift [-]. Default: 1.

Scale.LGAX

Optional. Scale factor of camber for F_x [-].Default: 1.

Scale.LCY

Optional. Scale factor of F_y shape factor [-]. Default: 1.

Scale.LMUY

Optional. Scale factor of F_y peak friction coefficient [-]. Default: 1.

Scale.LEY

Optional. Scale factor of F_y curvature factor [-]. Default: 1.

Scale.LKY

Optional. Scale factor of F_y cornering stiffness [-]. Default: 1.

Scale.LHY

Optional. Scale factor of F_y horizontal shift [-]. Default: 1.

Scale.LVY

Optional. Scale factor of F_y vertical shift [-]. Default: 1.

Scale.LGAY

Optional. Scale factor of camber for F_y [-].Default: 1.

Scale.LTR

Optional. Scale factor of peak of pneumatic trail [-]. Default: 1.

Scale.LRES

Optional. Scale factor for offset of residual torque [-]. Default: 1.

Scale.LGAZ

Optional. Scale factor of camber for M_z [-].Default: 1.

Scale.LMX

Optional. Scale factor of overturning couple [-]. Default: 1.

Scale.LVMX

Optional. Scale factor of M_x vertical shift [-]. Default: 1.

Scale.LMY

Optional. Scale factor of rolling resistance torque [-]. Default: 1.

Scale.LXAL

Optional. Scale factor of α influence on F_x [-]. Default: 1.

Scale.LYKA

Optional. Scale factor of κ influence on F_y [-]. Default: 1.

Scale.LVYKA

Optional. Scale factor of κ induced F_y [-]. Default: 1.

Scale.LS

Optional. Scale factor of moment arm of F_x [-]. Default: 1.

Scale.LSGKP

Optional. Scale factor of relaxation length of F_x [-]. Default: 1.

Scale.LSGAL

Optional. Scale factor of relaxation length of F_y [-]. Default: 1.

Scale.LGYR

Optional. Scale factor of gyroscopic torque [-]. Default: 1.

Scale.LMP

Optional. Scale factor of parking moment [-]. Default: 1.

Scale.LKYC

Optional. Scale factor of camber stiffness [-]. Default: 1.

Scale.LKZC

Optional. Scale factor of camber moment stiffness [-]. Default: 1.

Example All parameters are taken from the referenced ADAMS® tire property file `DT_Pacejca.tir`, in the directory `Data/Tire/Pacejka`.

The vertical stiffness from the referenced file is overwritten: The value 400000 N/m is used instead.

```
FileIdent =CarMaker-Tire-MF52 3
Description:
   ADAMS Pacejka parameter set
AdamsPropertyFile =Pacejka/DT_Pacejka.tir
Vertical.VERTICAL_STIFFNESS = 400000
```

17.5.4 Vertical force dependent on tire inflation pressure

The Magic Formula version 6.1 the vertical force F_z is calculated dependent on rolling tire radius R_Ω and tire inflation pressure.

The rolling free tire radius R_Ω is calculated using the following formula:

$$R_\Omega = R_0\left(q_{re0} + q_{v1}\left(\frac{\Omega R_0}{V_0}\right)^2\right) \quad \text{(EQ 265)}$$

The tire deflection ρ is the difference between the rolling free tire radius and the loaded radius R_l:

$$\rho = R_\Omega - R_l \quad \text{(EQ 266)}$$

The vertical tire force is then calculated with:

$$F_z = \left(1 + q_{v2}\frac{R_0}{V_0}|\Omega| - \left(\frac{q_{Fcx}F_x}{F_{z0}}\right)^2 - \left(\frac{q_{Fcy}F_y}{F_{z0}}\right)^2\right)\left(q_{Fz1}\frac{\rho}{R_0} + q_{Fz2}\left(\frac{\rho}{R_0}\right)^2\right)(1 + p_{Fz1}dp_i)F_{z0} \quad \text{(EQ 267)}$$

In the expressions for the effective rolling radius and contact patch dimensions the vertical stiffness dependent on tire inflation pressure is:

$$C_z = C_{z0}(1 + p_{Fz1}dp_i) \quad \text{(EQ 268)}$$

Model.LONGVL

Reference velocity V_0 [m/s].

Vertical.Q_RE0

Ratio of free tire radius with nominal tire radius [-].

Vertical.Q_V1

Tire radius increase with speed [-].

Vertical.Q_V2

Vertical stiffness increase with speed [-].

Vertical.Q_FZ1

Optional. Linear term in load versus deflection [-]. The default value is calculated as follows:

$$q_{Fz1} = \sqrt{\left(C_{z0}\frac{R_0}{F_{z0}}\right)^2 - 4q_{Fz2}} \quad \text{(EQ 269)}$$

Vertical.Q_FZ2

Quadratic term in load versus deflection [-].

Vertical.Q_FCX

Longitudinal force influence on vertical stiffness [-].

Vertical.Q_FCY

Lateral force influence on vertical stiffness [-].

Vertical.PFZ1

Pressure effect on vertical stiffness [-].

Vertical.FNOMIN

Nominal wheel load F_{z0} [N].

17.5.5 Effective tire rolling radius

The effective rolling radius is defined for a wheel the tire of which is uniform and rolls freely at constant speed over an even horizontal surface.It reads:

$$R_e = \frac{V_x}{\Omega} \quad \text{(EQ 270)}$$

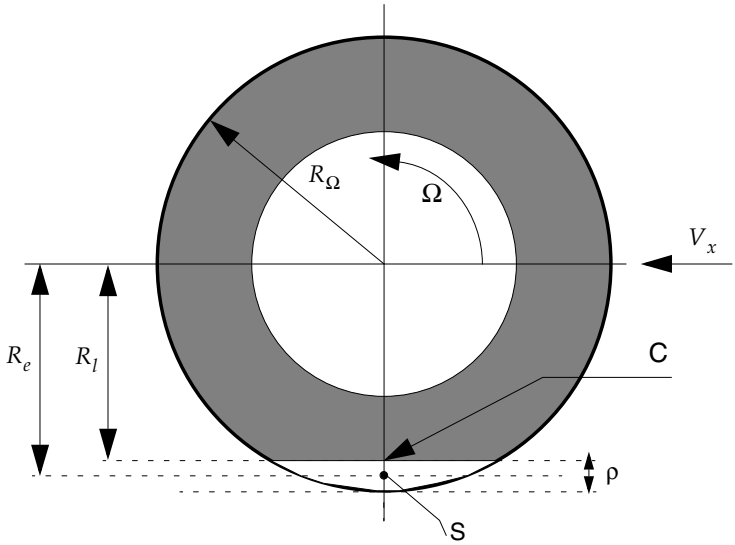


Figure 17.10: Effective rolling radius

In general the effective rolling radius will change with tire deflection ρ. Assuming the tire has a constant vertical stiffness C_z, the deflection is calculated with:

$$\rho = \frac{F_z}{C_z} \quad \text{(EQ 271)}$$

The effective rolling radius is estimated by:

$$R_e = R_\Omega - \rho_{F_{z0}}(F_{reff}\rho_d + D_{reff}\,\mathrm{atan}(B_{reff}\rho_d)) \quad \text{(EQ 272)}$$

$$\rho_{F_{z0}} = \frac{F_{z0}}{C_z} \qquad \rho_d = \frac{\rho}{\rho_{F_{z0}}} \quad \text{(EQ 273)}$$

In the Magic Formula version 5.2 R_Ω equals to the non-rolling free radius R_0.

Vertical.BREFF

Low load stiffness of effective rolling radius [-].

Vertical.DREFF

Peak value of effective rolling radius [-].

Vertical.FREFF

High load stiffness of effective rolling radius [-].

17.5.6 Slip computation

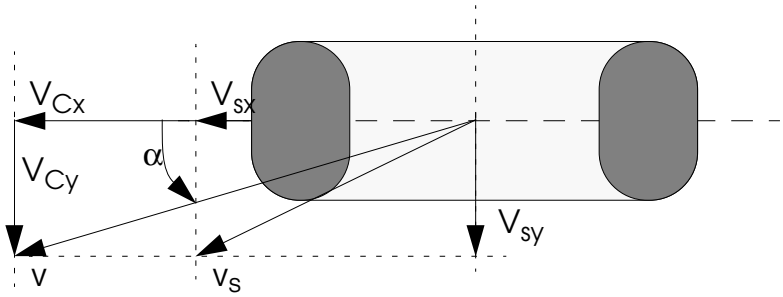


Figure 17.11: Slip computation

The longitudinal slip speed is defined as:

$$V_{sx} = V_{Cx} - \Omega R_e \quad \text{(EQ 274)}$$

which results in a definition of longitudinal slip:

$$\kappa = -\frac{V_{sx}}{V_{Cx}} \quad \text{(EQ 275)}$$

The lateral slip speed is defined as:

$$V_{sy} = V_{Cy} \quad \text{(EQ 276)}$$

which results in a definition of lateral slip:

$$\tan\alpha = \frac{V_{sy}}{|V_{Cx}|} \quad \text{(EQ 277)}$$

and a definition of sideslip angle:

$$\alpha = \operatorname{atan}\frac{V_{sy}}{|V_{Cx}|} \quad \text{(EQ 278)}$$

The rolling speed becomes:

$$V_r = R_e\Omega \quad \text{(EQ 279)}$$

In case the complete model including transient properties is used, the transient tire quantities κ' and α' (see (EQ 363), (EQ 364)) are used instead of the wheel slip quantities κ and α.

17.5.7 Longitudinal force (pure longitudinal slip):

$$F_{xo} = D_x \sin[C_x \operatorname{atan}\{B_x\kappa_x - E_x(B_x\kappa_x - \operatorname{atan}(B_x\kappa_x))\}] + S_{Vx} \quad \text{(EQ 280)}$$

$$\kappa_x = \kappa + S_{Hx} \quad \text{(EQ 281)}$$

$$C_x = p_{Cx1} \cdot \lambda_{Cx} \quad \text{(EQ 282)}$$

$$D_x = \mu_x \cdot F_z \cdot \zeta_1 \qquad (>0) \quad \text{(EQ 283)}$$

$$\mu_x = (p_{Dx1} + p_{Dx2} df_z) \cdot (1 - p_{Dx3}\gamma_x^2) \cdot \lambda_{\mu x} \quad (>0) \quad \text{with} \quad \gamma_x = \gamma \cdot \lambda_{\gamma x} \quad \text{(EQ 284)}$$

$$E_x = (p_{Ex1} + p_{Ex2} df_z + p_{Ex3} df_z^2) \cdot \{1 - p_{Ex4}\operatorname{sgn}(\kappa_x)\} \cdot \lambda_{Ex} \quad (\leq 1) \quad \text{(EQ 285)}$$

$$K_{x\kappa} = F_z \cdot (p_{Kx1} + p_{Kx2} df_z) \cdot \exp(p_{Kx3} df_z) \cdot \lambda_{Kx\kappa} \quad \text{(EQ 286)}$$

$$B_x = \frac{K_{xk}}{(C_x D_x + \varepsilon_x)} \quad \text{(EQ 287)}$$

$$S_{Hx} = (p_{Hx1} + p_{Hx2} df_z) \cdot \lambda_{Hx} \quad \text{(EQ 288)}$$

$$S_{Vx} = F_z \cdot (p_{Vx1} + p_{Vx2} df_z) \cdot \lambda_{Vx} \cdot \lambda_{\mu x} \cdot \zeta_1 \quad \text{(EQ 289)}$$

For the Magic Formula 6.1 version some equations above are extended with further expressions:

- In (EQ 284) μ_x is multiplied with the expression $(1 + p_{px3} dp_i + p_{px4} dp_i^2)$
- In (EQ 286) $K_{x\kappa}$ is multiplied with the expression $(1 + p_{px1} dp_i + p_{px2} dp_i^2)$

Longitudinal (pure slip)

Long.PCX1

Shape factor for longitudinal force [-].

Long.PDX1

Longitudinal friction μ_x at $F_{z_{nom}}$ [-].

Long.PDX2

Variation of friction μ_x with load [-].

Long.PDX3

Variation of friction μ_x with camber [-].

Long.PEX1

Longitudinal curvature at $F_{z_{nom}}$ [-].

Long.PEX2

Variation of curvature with load [-].

Long.PEX3

Variation of curvature with load squared [-].

Long.PEX4

Factor in curvature while driving [-].

Long.PKX1

Longitudinal slip stiffness at $F_{z_{nom}}$ [-].

Long.PKX2

Variation of slip stiffness with load [-].

Long.PKX3

Exponent in slip stiffness with load [-].

Long.PHX1

Horizontal shift at $F_{z_{nom}}$ [-].

Long.PHX2

Variation of shift with load [-].

Long.PVX1

Vertical shift at $F_{z_{nom}}$ [-].

Long.PVX2

Variation of shift with load [-].

Long.PPX1

Linear pressure effect on slip stiffness [-].

Long.PPX2

Quadratic pressure effect on slip stiffness [-].

Long.PPX3

Linear pressure effect on longitudinal friction [-].

Long.PPX4

Quadratic pressure effect on longitudinal friction [-].

17.5.8 Lateral force (pure sideslip):

$$F_{y0} = D_y \sin[C_y \operatorname{atan}\{B_y\alpha_y - E_y(B_y\alpha_y - \operatorname{atan}(B_y\alpha_y))\}] + S_{Vy} \quad \text{(EQ 290)}$$

$$\alpha_y = \alpha + S_{Hy} \quad \text{(EQ 291)}$$

$$C_y = p_{Cy1} \cdot \lambda_{Cy} \quad \text{(EQ 292)}$$

$$D_y = \mu_y \cdot F_z \cdot \zeta_2 \quad \text{(EQ 293)}$$

$$\mu_y = (p_{Dy1} + p_{Dy2} df_z) \cdot (1 - p_{Dy3}\gamma_y{}^2) \cdot \lambda_{\mu y} \quad \text{with } \gamma_y = \gamma \cdot \lambda_{\gamma y} \quad \text{(EQ 294)}$$

$$E_y = (p_{Ey1} + p_{Ey2} df_z) \cdot \{1 - (p_{Ey3} + p_{Ey4}\gamma_y)\operatorname{sgn}(\alpha_y)\} \cdot \lambda_{Ey} \quad (\le 1) \quad \text{(EQ 295)}$$

$$K_{y\alpha} = p_{Ky1} F_{z0}' \sin\left(2 \operatorname{atan}\left\{\frac{F_z}{p_{Ky2} F_{zo}'}\right\}\right) \cdot (1 - p_{Ky3}|\gamma_y|) \cdot \lambda_{Ky\alpha} \cdot \zeta_3 \quad \text{(EQ 296)}$$

$$B_y = \frac{K_{y\alpha}}{(C_y D_y + \varepsilon_y)} \quad \text{(EQ 297)}$$

$$S_{Hy} = (p_{Hy1} + p_{Hy2} df_z) \cdot \lambda_{Hy} + S_{Hy\gamma} \cdot \zeta_0 + \zeta_4 - 1 \quad \text{(EQ 298)}$$

$$S_{Hy\gamma} = p_{Hy3}\gamma_y \quad \text{(EQ 299)}$$

$$S_{Vy} = F_z \cdot (p_{Vy1} + p_{Vy2} df_z) \cdot \lambda_{Vy}\lambda_{\mu y} \cdot \zeta_2 + S_{Vy\gamma} \quad \text{(EQ 300)}$$

$$S_{Vy\gamma} = F_z \cdot (p_{Vy3} + p_{Vy4} df_z)\gamma_y \cdot \lambda_{Ky\gamma} \cdot \lambda_{\mu y} \cdot \zeta_2 \quad \text{(EQ 301)}$$

$$K_{y\gamma 0} = p_{Hy3} K_{y\alpha 0} + F_z(p_{Vy3} + p_{Vy4} df_z) \quad \text{(EQ 302)}$$

For the Magic Formula 6.1 version some equations above are modified or extended:

- In (EQ 294) μ_y is multiplied with the expression $(1 + p_{py3} dp_i + p_{py4} dp_i^2)$
- The (EQ 295) is extended to the equation:

$$E_y = (p_{Ey1} + p_{Ey2} df_z) \cdot \{1 + p_{Ey5}\gamma_y^2 - (p_{Ey3} + p_{Ey4}\gamma_y)\operatorname{sgn}(\alpha_y)\} \cdot \lambda_{Ey} \quad \text{(EQ 303)}$$

- The (EQ 296) is modified to the formula:

$$K_{y\alpha} = p_{Ky1} F_{z0}' \sin\left(p_{Ky4} \operatorname{atan}\left\{\frac{F_z}{(p_{Ky2} + p_{Ky5}\gamma_y^2)(1 + p_{py2} dp_i) F_{zo}'}\right\}\right) \cdot \quad \text{(EQ 304)}$$

$$(1 - p_{Ky3}|\gamma_y|)(1 + p_{py1} dp_i) \cdot \lambda_{Ky\alpha} \cdot \zeta_3$$

- The (EQ 302) is modified to the formula:

$$K_{y\gamma 0} = F_z \lambda_{Ky\gamma}(p_{Ky6} + p_{Ky7} df_z)(1 + p_{py5} dp_i) \quad \text{(EQ 305)}$$

- The (EQ 299) is modified to the formula:

$$S_{Hy\gamma} = \frac{K_{y\gamma 0} \cdot \gamma_y - S_{Vy\gamma}}{K_{y\alpha}'} \quad \text{(EQ 306)}$$

Lateral (pure slip)

Lat.PCY1

Shape factor for lateral forces [-].

Lat.PDY1

Lateral friction μ_y [-].

Lat.PDY2

Variation of friction μ_y with load [-].

Lat.PDY3

Variation of friction μ_y with squared camber [-].

Lat.PEY1

Lateral curvature at Fznom $F_{z_{nom}}$ [-].

Lat.PEY2

Variation of curvature with load [-].

Lat.PEY3

Zero order camber dependency of curvature [-].

Lat.PEY4

Variation of curvature with camber [-].

Lat.PEY5

Variation of curvature with squared camber [-].

Lat.PKY1

Maximum value of stiffness at $F_{z_{nom}}$ [-].

Lat.PKY2

Load at which K_{fy} reaches maximum value [-].

Lat.PKY3

Variation of $\frac{K_{fy}}{F_{z_{nom}}}$ with camber [-].

Lat.PKY4

Peak stiffness variation with squared camber [-].

Lat.PKY5

Lateral stiffness dependency with camber [-].

Lat.PKY6

Camber stiffness factor [-].

Lat.PKY7

Load dependency of camber stiffness factor [-].

Lat.PHY1

Horizontal shift at $F_{z_{nom}}$ [-].

Lat.PHY2

Variation of horizontal shift with load [-].

Lat.PHY3

Variation of horizontal shift with camber [-].

Lat.PVY1

Vertical shift at $F_{z_{nom}}$.

Lat.PVY2

Variation of vertical shift with load [-].

Lat.PVY3

Variation of vertical shift with camber [-].

Lat.PVY4

Variation of vertical shift with camber and load [-].

Lat.PPY1

Pressure effect on cornering stiffness magnitude [-].

Lat.PPY2

Pressure effect on location of cornering stiffness peak [-].

Lat.PPY3

Linear pressure effect on lateral friction [-].

Lat.PPY4

Quadratic pressure effect on lateral friction [-].

Lat.PPY5

Pressure effect on camber stiffness [-].

17.5.9 Aligning Torque (pure sideslip)

$$M_{z0} = M'_{z0} + M_{zr0} \quad \text{(EQ 307)}$$

$$M'_{z0} = -t_0 \cdot F_{y0} \quad \text{(EQ 308)}$$

$$t_0 = t(\alpha_t) = D_t \cos[C_t \operatorname{atan}\{B_t\alpha_t - E_t(B_t\alpha_t - \operatorname{atan}(B_t\alpha_t))\}] \cdot \cos\alpha \quad \text{(EQ 309)}$$

$$\alpha_t = \alpha + S_{Ht} \quad \text{(EQ 310)}$$

$$S_{Ht} = q_{Hz1} + q_{Hz2}df_z + (q_{Hz3} + q_{Hz4}df_z)\gamma_z \quad \text{with } \gamma_z = \gamma \cdot \lambda_{\gamma z} \quad \text{(EQ 311)}$$

$$M_{zr0} = M_{zr}(\alpha_r) = D_r \cos[C_r \operatorname{atan}(B_r\alpha_r)]\cos(\alpha) \quad \text{(EQ 312)}$$

$$C_r = \zeta_7 \quad \text{(EQ 313)}$$

$$\alpha_r = \alpha + S_{Hf}(= \alpha_f) \quad \text{(EQ 314)}$$

$$S_{Hf} = S_{Hy} + \frac{S_{Vy}}{K'_{y\alpha}} \quad \text{(EQ 315)}$$

$$K'_{y\alpha} = K_{y\alpha} + \varepsilon_K \quad \text{(EQ 316)}$$

$$B_t = (q_{Bz1} + q_{Bz2}df_z + q_{Bz3}df_z^2) \cdot (1 + q_{Bz4}\gamma_z + q_{Bz5}|\gamma_z|) \cdot \frac{\lambda_{Ky\alpha}}{\lambda_{\mu y}} \quad \text{(EQ 317)}$$

$$C_t = q_{Cz1} \quad \text{(EQ 318)}$$

$$D_{t0} = F_z \cdot \left(\frac{R_0}{F_{z0}}\right) \cdot (q_{Dz1} + q_{dz2}df) \cdot \lambda_t \quad \text{(EQ 319)}$$

$$D_t = D_{t0} \cdot (1 + q_{Dz3}\gamma_z + q_{Dz4}{\gamma_z}^2) \cdot \zeta_5 \quad \text{(EQ 320)}$$

$$E_t = (q_{Ez1} + q_{Ez2}df_z + q_{Ez3}df_z^2)\left\{1 + (q_{Ez4} + q_{Ez5}\gamma_z)\frac{2}{\pi}\operatorname{atan}(B_tC_t\alpha_t)\right\} \quad (\leq 1) \quad \text{(EQ 321)}$$

$$B_r = \left(q_{Bz9} \cdot \frac{\lambda_{Ky}}{\lambda_{\mu y}} + q_{Bz10}B_yC_y\right) \cdot \zeta_6 \quad \text{(EQ 322)}$$

$$D_r = F_zR_0\lambda_{\mu y}\{(q_{Dz6} + q_{Dz7}df_z)\lambda_{Mr}\zeta_2 + (q_{Dz8} + q_{Dz9}df_z)\gamma_z\zeta_0\} \cdot \zeta_8 - 1 \quad \text{(EQ 323)}$$

For the Magic Formula 6.1 version some equations above are extended with further expressions:

- In (EQ 308) F_{y0} is replaced by $F_{y0}|_{\gamma = 0,\, \varphi = 0}$
- In (EQ 320) D_t is multiplied with the expression $(1 - p_{pz1}dp_i)$
- The (EQ 323) is extended to the formula:

$$D_r = F_zR_0\lambda_{\mu y}\{(q_{Dz6} + q_{Dz7}df_z)\lambda_{Mr}\zeta_2 + (q_{Dz8} + q_{Dz9}df_z)(1 + p_{pz2}dp_i)\gamma_z\zeta_0\lambda_{Kz\gamma} + (q_{Dz10} + q_{Dz11}df_z)\gamma_z|\gamma_z|\lambda_{Kz\gamma}\zeta_0\} \cdot \zeta_8 - 1 \quad \text{(EQ 324)}$$

Aligning torque (pure slip)

Align.QBZ1

Trail slope factor for trail at $F_{z_{nom}}$ [-].

Align.QBZ2

Variation of slope with load [-].

Align.QBZ3

Variation of slope with load squared [-].

Align.QBZ4

Variation of slope with camber [-].

Align.QBZ5

Variation of slope with absolute camber [-].

Align.QBZ9

Slope factor of residual torque [-].

Align.QBZ10

Slope factor of residual torque [-].

Align.QCZ1

Shape factor for pneumatic trail [-].

Align.QDZ1

Peak trail [-].

Align.QDZ2

Variation of peak trail with load [-].

Align.QDZ3

Variation of peak trail with camber [-].

Align.QDZ4

Variation of peak trail with camber squared [-].

Align.QDZ6

Peak residual torque [-].

Align.QDZ7

Variation of peak residual torque factor with load [-].

Align.QDZ8

Variation of peak residual torque factor with camber [-].

Align.QDZ9

Variation of peak residual torque factor with camber and load [-].

Align.QDZ10

Variation of peak factor with camber squared [-].

Align.QDZ11

Variation of peak factor with camber squared and load [-].

Align.QEZ1

Trail curvature at $F_{z_{nom}}$ [-].

Align.QEZ2

Variation of curvature with load [-].

Align.QEZ3

Variation of curvature with load squared [-].

Align.QEZ4

Variation of curvature with $\mathrm{sgn}(\alpha_t)$ [-].

Align.QEZ5

Variation of curvature with camber and $\mathrm{sgn}(\alpha_t)$ [-].

Align.QHZ1

Trail horizontal shift at $F_{z_{nom}}$ [-].

Align.QHZ2

Variation of shift with load [-].

Align.QHZ3

Variation of shift with camber [-]-

Align.QHZ4

Variation of shift with camber and load [-].

Align.PPZ1

Linear pressure effect on pneumatic trail [-].

Align.PPZ2

Linear pressure effect on residual aligning torque [-].

17.5.10 Longitudinal Force (combined slip)

$$F_x = G_{x\alpha} \cdot F_{xo} \quad \text{(EQ 325)}$$

$$G_{x\alpha} = \frac{\cos[C_{x\alpha}\operatorname{atan}\{B_{x\alpha}\alpha_S - E_{x\alpha}(B_{x\alpha}\alpha_S - \operatorname{atan}(B_{x\alpha}\alpha_S))\}]}{G_{x\alpha o}} \quad (>0) \quad \text{(EQ 326)}$$

$$G_{x\alpha 0} = \cos[C_{x\alpha}\operatorname{atan}\{B_{x\alpha}S_{Hx\alpha} - E_{x\alpha}(B_{x\alpha}S_{Hx\alpha} - \operatorname{atan}(B_{x\alpha}S_{Hx\alpha}))\}] \quad \text{(EQ 327)}$$

$$\alpha_S = \alpha + S_{Hx\alpha} \quad \text{(EQ 328)}$$

$$B_{x\alpha} = r_{Bx1}\cos([\operatorname{atan}(r_{Bx2}\kappa)] \cdot \lambda_{x\alpha}) \quad \text{(EQ 329)}$$

$$C_{x\alpha} = r_{Cx1} \quad \text{(EQ 330)}$$

$$E_{x\alpha} = r_{Ex1} + r_{Ex2}df_z \quad \text{(EQ 331)}$$

$$S_{Hx\alpha} = r_{Hx1} \quad \text{(EQ 332)}$$

For the Magic Formula 6.1 version in (EQ 329) instead of r_{Bx1} the expression $r_{Bx1} + r_{Bx3}\gamma^2$ is used.

Longitudinal (combine)

Long.RBX1

Slope factor for combined slip reduction [-].

Long.RBX2

Variation of slope F_x reduction with κ [-].

Long.RBX3

Influence of camber on stiffness for F_x combined [-].

Long.RCX1

Shape factor for combined slip F_x reduction [-].

Long.REX1

Optional. Curvature factor of combined F_x [-]. Default: 0.

Long.PEX2

Optional. Curvature factor of combined F_x with load [-].Default: 0.

Long.PHX1

Shift factor for combined slip F_x reduction.

17.5.11 Lateral force (combined slip)

$$F_y = G_{y\kappa} \cdot F_{yo} + S_{Vy\kappa} \quad \text{(EQ 333)}$$

$$G_{y\kappa} = \frac{\cos[C_{y\kappa}\operatorname{atan}\{B_{y\kappa}\kappa_S - E_{y\kappa}(B_{y\kappa}\kappa_S - \operatorname{atan}(B_{y\kappa}\kappa_S))\}]}{\cos[C_{y\kappa}\operatorname{atan}\{B_{y\kappa}S_{Hy\kappa} - E_{y\kappa}(B_{y\kappa}S_{Hy\kappa} - \operatorname{atan}(B_{y\kappa}S_{Hy\kappa}))\}]} \quad \text{(EQ 334)}$$

$$\kappa_S = \kappa + S_{Hy\kappa} \quad \text{(EQ 335)}$$

$$B_{y\kappa} = r_{By1}\cos[\operatorname{atan}\{r_{By2}(\alpha - r_{By3})\}] \cdot \lambda_{y\kappa} \quad \text{(EQ 336)}$$

$$C_{y\kappa} = r_{Cy1} \quad \text{(EQ 337)}$$

$$E_{y\kappa} = r_{Ey1} + r_{Ey2}df_z \qquad (\leq 1) \quad \text{(EQ 338)}$$

$$S_{Hy\kappa} = r_{Hy1} + r_{Hy2}df_z \quad \text{(EQ 339)}$$

$$S_{Vy\kappa} = D_{Vy\kappa}\sin[r_{Vy5}\operatorname{atan}(r_{Vy6}\kappa)]\lambda_{Vy\kappa} \quad \text{(EQ 340)}$$

$$D_{Vy\kappa} = \mu_y F_z \cdot (r_{Vy1} + r_{Vy2}df_z + r_{Vy3}\gamma) \cdot \cos(\operatorname{atan}(r_{Vy4}\alpha)) \cdot \zeta_2 \quad \text{(EQ 341)}$$

For the Magic Formula 6.1 version in (EQ 336) for r_{By1} the expression $r_{By1} + r_{By4}\gamma^2$ is used.

Lateral (combined slip)

Lat.RBY1

Slope factor for combined F_y reduction [-].

Lat.RBY2

Variation of slope F_y reduction with alpha [-].

Lat.RBY3

Shift term for alpha in slope F_y reduction [-].

Lat.RBY4

Influence of camber on stiffness of F_y combined [-].

Lat.RCY1

Shape factor for combined F_y reduction [-].

Lat.REY1

Optional. Curvature factor of combined F_y [-]. Default: 0.

Lat.REY2

Optional. Curvature factor of combined F_y with load [-].Default: 0.

Lat.RHY1

Shift factor for combined F_y reduction [-].

Lat.RHY2

Optional. Shift factor for combined F_y reduction with load [-]. Default: 0.

Lat.RVY1

κ induced side force at $F_{z_{nom}}$ [-].

Lat.RVY2

Variation of κ induced side force with load [-].

Lat.RVY3

Variation of κ induced side force with camber [-].

Lat.RVY4

Variation of κ induced side force with α [-].

Lat.RVY5

Variation of κ induced side force with κ [-].

Lat.RVY6

Variation of κ induced side force with $\mathrm{atan}(\kappa)$ [-].

17.5.12 Aligning Torque (combined slip)

$$M_z = M_z' + M_{zr} + s \cdot F_x \quad \text{(EQ 342)}$$

$$M_z' = -t \cdot F_y' \quad \text{(EQ 343)}$$

$$t = t(\alpha_{t,eq}) = D_t \cos[C_t \operatorname{atan}\{B_t \alpha_{t,eq} - E_t(B_t \alpha_{t,eq} - \operatorname{atan}(B_t \alpha_{t,eq}))\}] \cdot \cos\alpha \quad \text{(EQ 344)}$$

$$F_y' = F_y - S_{Vy\kappa} \quad \text{(EQ 345)}$$

$$M_{zr} = M_{zr}(\alpha_{r,eq}) = D_r \cos[C_r \operatorname{atan}(B_r \alpha_{r,eq})] \cdot \cos\alpha \quad \text{(EQ 346)}$$

$$s = R_0 \cdot \left\{ s_{sz1} + s_{sz2}\left(\frac{F_y}{F_{z0}}\right) + (s_{sz3} + s_{sz4} df_z)\gamma \right\} \cdot \lambda_s \quad \text{(EQ 347)}$$

$$\alpha_{t,eg} = \operatorname{atan}\sqrt{\tan\alpha_t^2 + \left(\frac{K_{x\kappa}}{K_{y\alpha}'}\right)^2 \kappa^2} \cdot \operatorname{sgn}(\alpha_t) \quad \text{(EQ 348)}$$

$$\alpha_{r,eg} = \operatorname{atan}\sqrt{\tan\alpha_r^2 + \left(\frac{K_{x\kappa}}{K_{y\alpha}'}\right)^2 \kappa^2} \cdot \operatorname{sgn}(\alpha_r) \quad \text{(EQ 349)}$$

For the Magic Formula 6.1 version in (EQ 345) F_y' is calculated in following way:

$$F_y' = F_{y0}\big|_{\gamma=0,\,\varphi=0} \cdot G_{y\kappa}\big|_{\gamma=0,\,\varphi=0} \quad \text{(EQ 350)}$$

Aligning Torque (combined slip)

Align.SSZ1

Nominal value of $\frac{s}{R_o}$ effect of F_x on M_z [-].

Align.SSZ2

Variation of distance $\frac{s}{R_o}$ with $\frac{F_y}{F_{z_{nom}}}$ [-].

Align.SSZ3

Variation of distance $\frac{s}{R_o}$ with camber [-].

Align.SSZ4

Variation of distance $\frac{s}{R_o}$ with load and camber [-].

17.5.13 Overturning Couple

$$M_x = F_z R_0 \left(q_{sx1} \lambda_{VMx} + \left(-q_{sx2}\gamma + q_{sx3}\frac{F_y}{F_{z0}} \right) \lambda_{Mx} \right) \quad \text{(EQ 351)}$$

For the Magic Formula 6.1 version the (EQ 351) is extended to the formula:

$$\begin{aligned} M_x = {} & \lambda_{Mx} F_z R_0 \Bigg\{ q_{sx1}\lambda_{VMx} - q_{sx2}\gamma(1 + p_{pmx1} dp_i) - q_{sx12}\gamma|\gamma| + q_{sx3}\frac{F_y}{F_{z0}} \\ & + q_{sx4}\cos\left[q_{sx5}\operatorname{atan}\left(\left(q_{sx6}\frac{F_z}{F_{z0}} \right)^2 \right) \right] \cdot \sin\left[q_{sx7}\gamma + q_{sx8}\operatorname{atan}\left(q_{sx9}\frac{F_y}{F_{z0}} \right) \right] \\ & + q_{sx10}\operatorname{atan}\left(q_{sx11}\frac{F_z}{F_{z0}} \right)\gamma \Bigg\} + \lambda_{Mx} F_y R_0 \{ q_{sx13} + q_{sx14}|\gamma| \} \end{aligned} \quad \text{(EQ 352)}$$

Overturning Couple

OverTurn.QSX1

Lateral force induced overturning couple [-].

OverTurn.QSX2

Camber induced overturning couple [-].

OverTurn.QSX3

F_y induced overturning couple [-].

OverTurn.QSX4

Mixed load, lateral force and camber on M_x [-].

OverTurn.QSX5

Load effect on M_x with lateral force and camber [-].

OverTurn.QSX6

B-factor of load with M_x [-].

OverTurn.QSX7

Camber with load on M_x [-].

OverTurn.QSX8

Lateral force with load on M_x [-].

OverTurn.QSX9

B-factor of lateral force with load on M_x [-].

OverTurn.QSX10

Vertical force with camber on M_x [-].

OverTurn.QSX11

B-factor of vertical force with camber on M_x [-].

OverTurn.QSX12

Camber squared induced M_x [-].

OverTurn.QSX13

Lateral force induced M_x [-].

OverTurn.QSX14

Lateral force induced M_x with camber [-].

OverTurn.PPMX1

Influence of inflation pressure on M_x [-].

17.5.14 Rolling Resistance Torque

$$M_y = -F_z R_0 \cdot \left\{ q_{sy1} + q_{sy2}\frac{F_x}{F_{z0}} + q_{sy3}\left|\frac{V_x}{V_{ref}}\right| + q_{sy4}\left(\frac{V_x}{V_{ref}}\right)^4 \right\} \cdot \lambda_{My} \quad \text{(EQ 353)}$$

where V_{ref} means measurement speed.

If q_{sy1} and q_{sy2} are both zero, then the following formula is used:

$$M_y = -R_0(S_{Vx} + K_x \cdot S_{Hx}) \quad \text{(EQ 354)}$$

For the Magic Formula 6.1 version the (EQ 353) is extended to the formula:

$$M_y = -\lambda_{My} F_{z0} R_0 \cdot \left\{ q_{sy1} + q_{sy2}\frac{F_x}{F_{z0}} + q_{sy3}\left|\frac{V_x}{V_{ref}}\right| + q_{sy4}\left(\frac{V_x}{V_{ref}}\right)^4 + q_{sy5}\gamma^2 + q_{sy6}\frac{F_z}{F_{z0}}\gamma^2 \right\} \cdot \left(\frac{F_z}{F_{z0}}\right)^{q_{sy7}} \left(\frac{P}{P_0}\right)^{q_{sy8}} \quad \text{(EQ 355)}$$

Roll Resistance Torque

Roll.QSY1

Rolling resistance torque coefficient [-].

Roll.QSY2

Rolling resistance torque depending on F_x [-].

Roll.QSY3

Rolling resistance torque depending on speed [-].

Roll.QSY4

Rolling resistance torque depending on $speed^4$ [-].

Roll.QSY5

Rolling resistance torque depending on camber squared [-].

Roll.QSY6

Rolling resistance torque depending on load and camber squared [-].

Roll.QSY7

Rolling resistance torque coefficient load dependency [-].

Roll.QSY8

Rolling resistance torque coefficient pressure dependency [-].

Roll.LONGVL

Optional. Measurement speed. Default:10 [m/s].

Additional rolling resistance models

Besides the rolling resistance model according to MFTIRE 5.2, the user has the possibility to use an alternative rolling resistance model.

Roll.Kind = ***KindStr***

The following alternative rolling resistance models are supported:

KindStr	Description
TrqLoadVel	A torque along wheel spin axis, against rotation based on load force, pressure and velocity according SAE J2452.

Parameters for the model 'TrqLoadVel'

Roll.Press = ***value***
Roll.alpha = ***value***
Roll.beta = ***value***
Roll.a = ***value***
Roll.b = ***value***
Roll.c = ***value***

For details please refer to section 17.4.1 'Rolling Resistance'.

17.5.15 Transient Behavior

For transient effects slip speeds instead of α and κ are used. First-order lags for tire longitudinal and lateral deformations u and v are introduced through relaxation length σ_k and σ_a:

$$\sigma_\kappa \frac{du}{dt} + |V_{Cx}|u = -\sigma_\kappa V_{Sx} \quad \text{(EQ 356)}$$

$$\sigma_\alpha \frac{dv}{dt} + |V_{Cx}|v = \sigma_\alpha V_{Sy} \quad \text{(EQ 357)}$$

These differential equations are based on the assumption that the contact point is not sliding.The relaxation lengths are only functions of vertical load and camber angle in conformity to ADAMS implementation of MF TIRE 5.2:

$$\sigma_\kappa = F_z \cdot (p_{Tx1} + p_{Tx2} df_z) \cdot \exp(-p_{Tx3} df_z) \cdot \frac{R_0}{F_{zo}} \cdot \lambda_{\sigma\kappa} \quad \text{(EQ 358)}$$

$$\sigma_\alpha = p_{Ty1} \sin\left[2\,\mathrm{atan}\left\{\frac{F_z}{p_{Ty2} F_{z0}'}\right\}\right] \cdot (1 - p_{Ky3}|\gamma|) \cdot R_0 \lambda_{F_{zo}} \lambda_{\sigma\alpha} \quad \text{(EQ 359)}$$

For the Magic Formula 6.1 version the (EQ 358) and (EQ 359) are modified to the following equations:

$$\sigma_\kappa = \frac{K_{xk}}{C_x} \text{ and } \sigma_\alpha = \frac{K_{y\alpha}}{C_y} \quad \text{(EQ 360)}$$

with overall longitudinal and lateral stiffness C_x and C_y:

$$C_x = C_{x0}(1 + p_{cfx1} df_z + p_{cfx2} df_z^2)(1 + p_{cfx3} dp_i) \quad \text{(EQ 361)}$$

$$C_y = C_{y0}(1 + p_{cfy1} df_z + p_{cfy2} df_z^2)(1 + p_{cfy3} dp_i) \quad \text{(EQ 362)}$$

The practical tire deformation quantities κ' and α' are subsequently used instead of arguments κ and α in the equations. They are defined as follows:

$$\kappa' = \frac{u}{\sigma_\kappa} \mathrm{sgn}(V_{Cx}) \quad \text{(EQ 363)}$$

$$\tan\alpha' = \frac{v}{\sigma_\alpha} \quad \text{(EQ 364)}$$

Starting from (EQ 356) and (EQ 357) the transient behavior could be described with a low pass filter. The time constant reads:

$$T = \frac{\sigma_\kappa}{|V_x|} \quad \text{(EQ 365)}$$

for the longitudinal action and

$$T = \frac{\sigma_\alpha}{|V_x|} \quad \text{(EQ 366)}$$

for lateral action.

Transient Behavior

Long.PTX1

Relaxation length at $F_{z_{nom}}$ [-].

Long.PTX2

Variation of relaxation length with load [-].

Long.PTX3

Variation of relaxation length with exponent of load [-].

Lat.PTY1

Peak value of relaxation length [-].

Lat.PTY2

Shape factor for relaxation length [-].

Structur.LONGITUDINAL_STIFFNESS

Tire overall longitudinal stiffness C_{x0} [N/m].

Structur.LATERAL_STIFFNESS

Tire overall lateral stiffness C_{y0} [N/m].

Structur.PCFX1

Tire overall longitudinal stiffness vertical deflection dependency linear term [-].

Structur.PCFX2

Tire overall longitudinal stiffness vertical deflection dependency quadratic term [-].

Structur.PCFX3

Tire overall longitudinal stiffness pressure dependency [-].

Structur.PCFY1

Tire overall lateral stiffness vertical deflection dependency linear term [-].

Structur.PCFY2

Tire overall lateral stiffness vertical deflection dependency quadratic term [-].

Structur.PCFY3

Tire overall lateral stiffness pressure dependency [-].

17.5.16 Turn Slip and Parking

Using the USE_MODE 5 or 15, described in section 17.5.2 'Tire Description', the forces and torques calculation considers the turn slip (turn spin) as an additional input. In this case the reductions factors ζ_i, appearing in the equations, are not set to one as customary, but determined as function of turn slip or turn spin.

Turn Slip and Turn Spin

The turn slip is defined as:

$$\varphi_t = \frac{\dot{\psi}}{|V_x'|} \quad \text{(EQ 367)}$$

with the tire turning velocity $\dot{\psi}$ and the singularity protected velocity $V_x' = max(V_x, V_{low})$.

The total tire spin is defined as:

$$\varphi = \frac{(\dot{\psi} - \{1 - \varepsilon_\gamma\} \cdot \Omega \cdot \sin\gamma)}{|V_x'|} \quad \text{(EQ 368)}$$

with rim rotation speed Ω, inclination angle γ and reduction factor ε_γ:

$$\varepsilon_\gamma = p_{\varepsilon\gamma\varphi 1}(1 + p_{\varepsilon\gamma\varphi 2} df_z) \quad \text{(EQ 369)}$$

Turn.PECP1

Camber spin reduction factor in camber stiffness [-].

Turn.PECP2

Camber spin reduction factor varying with load in camber stiffness [-].

Reduction factor for Longitudinal force

The reduction factor ζ_1 from the equations (EQ 283) and (EQ 289) is given by:

$$\zeta_1 = \cos[\operatorname{atan}(B_{x\varphi} R_0 \varphi)] \quad \text{(EQ 370)}$$

$$B_{x\varphi} = p_{Dx\varphi 1}(1 + p_{Dx\varphi 2} df_z) \cdot \cos[\operatorname{atan}(p_{Dx\varphi 3}\kappa)] \quad \text{(EQ 371)}$$

Turn.PDXP1

Peak F_x reduction due to spin [-].

Turn.PDXP2

Peak F_x reduction due to spin with varying load [-].

Turn.PDXP3

Peak F_x reduction due to spin with kappa [-].

Reduction factors for Lateral force

The reduction factor ζ_2 from the equations (EQ 293), (EQ 300) and (EQ 301) is given by:

$$\zeta_2 = \cos[\operatorname{atan}(B_{y\varphi}R_0|\varphi| + p_{Dy\varphi 4}\sqrt{R_0|\varphi|})] \quad \text{(EQ 372)}$$

$$B_{y\varphi} = p_{Dy\varphi 1}(1 + p_{Dy\varphi 2}df_z) \cdot \cos[\operatorname{atan}(p_{Dy\varphi 3}\tan\alpha)] \quad \text{(EQ 373)}$$

The reduction factor ζ_3 from the equation (EQ 296) is defined as follows:

$$\zeta_3 = \cos[\operatorname{atan}(p_{Ky\varphi 1}R_0^2\varphi^2)] \quad \text{(EQ 374)}$$

The reduction factor ζ_0 from the equation (EQ 298) is zero.

The reduction factor ζ_4 from the equation (EQ 298) reads:

$$\zeta_4 = 1 + S_{Hy\varphi} - \frac{S_{Vy\gamma}}{K'_{y\alpha}} \quad \text{(EQ 375)}$$

$$S_{Vy\gamma} = F_z \cdot (p_{Vy3} + p_{Vy4}df_z)\gamma_y \cdot \lambda_{\mu y}\lambda_{Ky\gamma} \cdot \zeta_2 \quad \text{(EQ 376)}$$

$$S_{Hy\varphi} = D_{Hy\varphi}\sin[C_{Hy\varphi}\operatorname{atan}\{B_{Hy\varphi}R_0\varphi - E_{Hy\varphi}(B_{Hy\varphi}R_0\varphi - \operatorname{atan}(B_{Hy\varphi}R_0\varphi))\}] \quad \text{(EQ 377)}$$

$$C_{Hy\varphi} = p_{Hy\varphi 1} \quad \text{(EQ 378)}$$

$$D_{Hy\varphi} = (p_{Hy\varphi 2} + p_{Hy\varphi 3}df_z) \cdot sign(V_x) \quad \text{(EQ 379)}$$

$$E_{Hy\varphi} = p_{Hy\varphi 4} \quad \text{(EQ 380)}$$

$$K_{yR\varphi 0} = \frac{K_{y\gamma 0}}{1 - \varepsilon_\gamma} \quad \text{(EQ 381)}$$

$$B_{Hy\varphi} = \frac{K_{yR\varphi 0}}{C_{Hy\varphi}D_{Hy\varphi}K'_{y\alpha 0}} \quad \text{(EQ 382)}$$

Turn.PDYP1

Peak F_y reduction due to spin [-].

Turn.PDYP2

Peak F_y reduction due to spin with varying load [-].

Turn.PDYP3

Peak F_y reduction due to spin with alpha [-].

Turn.PDYP4

Peak F_y reduction due to square root of spin [-].

Turn.PKYP1

Cornering stiffness $K_{y\alpha}$ reduction due to spin [-].

Turn.PHYP1

Lateral shift $S_{Hy\varphi}$ shape factor [-].

Turn.PHYP2

Lateral shift $S_{Hy\varphi}$ peak factor [-].

Turn.PHYP3

Lateral shift $S_{Hy\varphi}$ peak factor varying with load [-].

Turn.PHYP4

Lateral shift $S_{Hy\varphi}$ curvature factor [-].

Reduction factors for Aligning Torque

The reduction factor ζ_5 from the equation (EQ 320) reads:

$$\zeta_5 = \cos[\operatorname{atan}(q_{Dt\varphi 1} R_0 \varphi)] \quad \text{(EQ 383)}$$

The reduction factor ζ_6 from the equation (EQ 322) is defined as follows:

$$\zeta_6 = \cos[\operatorname{atan}(q_{Br\varphi 1} R_0 \varphi)] \quad \text{(EQ 384)}$$

The reduction factor ζ_8 from the equation (EQ 323) is given by:

$$\zeta_8 = 1 + D_{r\varphi} \quad \text{(EQ 385)}$$

$$D_{r\varphi} = D_{Dr\varphi} \sin[C_{Dr\varphi} \operatorname{atan}\{B_{Dr\varphi} R_0 \varphi - E_{Dr\varphi}(B_{Dr\varphi} R_0 \varphi - \operatorname{atan}(B_{Dr\varphi} R_0 \varphi))\}] \quad \text{(EQ 386)}$$

$$C_{Dr\varphi} = q_{Dr\varphi 1} \quad \text{(EQ 387)}$$

$$E_{Dr\varphi} = q_{Dr\varphi 2} \quad \text{(EQ 388)}$$

$$M_{z\varphi\infty} = q_{Cr\varphi 1} \mu_y R_0 F_z \sqrt{\frac{F_z}{F_{z0}'}} \cdot \lambda_{M\varphi} \quad \text{(EQ 389)}$$

$$D_{Dr\varphi} = \frac{M_{z\varphi\infty}}{\sin\left(\frac{\pi}{2} C_{Dr\varphi}\right)} \quad \text{(EQ 390)}$$

$$K_{z\gamma r0} = F_z R_0 (q_{Dz8} + q_{Dz9} df_z + (q_{Dz10} + q_{Dz11} df_z)|\gamma_z|) \cdot \lambda_{Kz\gamma} \quad \text{(EQ 391)}$$

$$B_{Dr\varphi} = \frac{K_{z\gamma r0}}{C_{Dr\varphi} D_{Dr\varphi}(1 - \varepsilon_\gamma)} \quad \text{(EQ 392)}$$

The reduction factor ζ_7 from the equation (EQ 313) is given by:

$$M_{z\varphi 90} = M_{z\varphi\infty} \frac{2}{\pi} \cdot a\tan(q_{Cr\varphi 2} R_0 |\varphi_t|) \cdot G_{y\kappa} \quad \text{(EQ 393)}$$

$$\zeta_7 = \frac{2}{\pi} a\cos(M_{z\varphi 90} / |D_{r\varphi}|) \quad \text{(EQ 394)}$$

Turn.QDTP1

Pneumatic trail D_t reduction factor due to spin [-].

Turn.QBRP1

Residual (spin) torque reduction factor due to spin [-].

Turn.QCRP1

Turning moment at constant turning and zero forward speed parameter [-].

Turn.QCRP2

Turn slip moment (at alpha=90deg) parameter for increase with spin [-].

Turn.QDRP1

Turn slip moment peak magnitude [-].

Turn.QDRP2

Turn slip moment peak position [-].

Transient Behavior of Turn Slip

In the same manner like for α and κ, the transient turn slip and turn spin φ' are modelled with a first-order lag:

$$\sigma_\varphi \frac{dw}{dt} + |V_{Cx}| w = \sigma_\varphi \dot{\psi} \quad \text{(EQ 395)}$$

$$\varphi' = \frac{w}{\sigma_\varphi} \quad \text{(EQ 396)}$$

with the relaxation σ_φ, which is simplified to the half contact length a:

$$a = p_{A1} R_0 \left(\frac{\rho}{R_0} + p_{A2} \sqrt{\frac{\rho}{R_0}} \right) \quad \text{(EQ 397)}$$

For the Magic Formula 6.1 version the (EQ 397) is modified to the formula:

$$a = R_0 \left(q_{ra2} \frac{F_z}{C_z R_0} + q_{ra1} \sqrt{\frac{F_z}{C_z R_0}} \right) \quad \text{(EQ 398)}$$

Contact.PA1
Contact.Q_RA2

Half contact length with vertical tire deflection [-].

Contact.PA2
Contact.Q_RA1

Half contact length with square root of vertical tire deflection [-].

17.5.17 Gyroscopic Couple

Due to the tire inertia acting about the vertical axis an offset is added to the aligning torque computed with equation (EQ 342):

$$M_{z,\,gyr} = c_{gyr} m_{Belt} V_{rl} \frac{dv}{dt} \cos[\operatorname{atan}(B_r \alpha_{r,\,eq})] \quad \text{(EQ 399)}$$

with parameter

$$c_{gyr} = q_{Tz1} \cdot \lambda_{gyr} \quad \text{(EQ 400)}$$

and

$$\cos[\operatorname{atan}(B_r \alpha_{r,\,eq})] = 1 \quad \text{(EQ 401)}$$

for pure cornering condition.

The total aligning torque becomes now:

$$M_z = M_z' + M_{z,\,gyr} \quad \text{(EQ 402)}$$

Gyroscopic Couple

Align.QTZ1

Gyroscopic torque constant [-].

Align.MBELT

Belt mass of the wheel [Kg].

17.5.18 Friction coefficient

If the tire parameters have been detemined at friction conditions $\mu_{Measure}$ which differ from the current friction condition at road μ_{Road}, an effective friction coefficient will be used for the tire computation. Additionally the friction coefficient is multiplied with the scale factors $\lambda_{\mu x}$ and $\lambda_{\mu y}$.With the current friction condition for the road patch the effective friction value used for the computation reads:

$$\mu_{eff\,x,\,y} = \mu_{Road} \cdot \lambda_{\mu x,\,y} / \mu_{Measure} \quad \text{(EQ 403)}$$

This value is used instead of $\lambda_{\mu x}$ respectively $\lambda_{\mu y}$.

17.5.19 User Accessible Quantities

Please refer to section 24.10.2 'Tire Model Magic Formula 5.2 and 6.1'.

17.6 Tire Model "Tame Tire"

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. For details please refer to the *TameTire User Manual.*
Supported platforms are linux32, linux64, win32, win64 and xeno.

SolverMode

Specifies the solver mode of the model:

- 1: mechanical model is used (no thermal effects considered)
- 2: thermo-mechanical model is used (thermal effects considered). Default.

CalcOpt

Optional. Specifies the calculation accuracy:

- 1: slowest and most accurate. Default.
- 2: trade-off between speed and accuracy.
- 3: fastest and less accuracy. Recommended for HIL simulations.

Load Force

InfPress

Specifies the tire inflation pressure [bar]. Default: 2.0.

Radial.Stiffness

Radial/vertical stiffness of tire [N/m].

Radial.Damping

Radial/vertical damping coefficient of tire [N/m/s].

Stand Still

StandStill.vMax

Boundary for switching between stand still model and normal computation [m/s].
Default: 0.25.

StandStill.cLong = ***Stiffness***
StandStill.cSide = ***Stiffness***

Stiffness for stand still model [-]. Default: 0.01, 0.2.

Initial Temperature

Init.TempCore

Specifies the initial core temperature of the compound [° C]. Default: 20.

Init.TempSurf

Specifies the initial surface temperature of the compound [° C]. Default: 20.

Init.TempTrack

Specifies the initial track temperature [° C]. Default: 20.

Scaling Factors

Scale.LongSlipStiff

Scaling factor of the tread shearing in the longitudinal direction []. Default: 1.0.

Scale.LatSlipStiff

Scaling factor of the tread shearing in the lateral direction []. Default: 1.0.

Scale.LongStiff

Scaling factor of the tire rigidity in the longitudinal direction []. Default: 1.0.

Scale.LatStiff

Scaling factor of the tire rigidity in the lateral direction []. Default: 1.0.

Scale.VertStiff

Scaling factor of the tire rigidity in the vertical direction []. Default: 1.0.

Scale.BeltCompl

Scaling factor of the belt compliance []. Default: 1.0.

Scale.TorsStiff

Scaling factor of the torsional stiffness []. Default: 1.0.

Scale.TreadDepth

Scaling factor of the tread depth []. Default: 1.0.

Scale.GlobalFric

Scaling factor of the global friction level between the tread and the track []. Default: 1.0.

Validation Limits

Limit.IncAng = *min max*

Specifies the validation limitation for the inclination angle [rad]. Default: -0.1 0.1.

Limit.Deflec = *min max*

Specifies the validation limitation for the tire deflection [m]. Default: 0 0.1.

Limit.TempSurf = *min max*

Specifies the validation limitation of the compound surface temperature [° C].
Default: 10 200.

Limit.TempCoreAvg = *min max*

Specifies the validation limitation of the compound average core temperature [° C].
Default: 10 100.

Misc

Side= *side*

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.
Possible values are "left" and "right".

Data: Text

This parameter contains encoded model coefficients.

17.6.1 User Accessible Quantities

Please refer to section 24.10.1 'Tire Model Tame Tire'.

17.7 Tire Model "MF-Tyre/MF-Swift 6.2"

To use the tire model *MF-Tire/MF-Swift 6.2* by TASS International, a corresponding dynamic library will be loaded, which is included in the CarMaker installation (currently only windows versions supported). The path to the library can also be set in the *SimParameters* file, if another library should be used. For use of MF-Swift options a separate license from TASS international is required. For details please refer to section 3.1.8 'Tire Model MF-Tyre/MF-Swift'.

AdamsPropertyFile = *PropertyFile*

Link to an ADAMS® tire property file. Its path is relative to directory of the tire infofile.

Supported FILE_TYPE:	"tir"
Supported FILE_VERSION:	2 or 3
Supported FILE_FORMAT:	"ASCII"
Supported PROPERTY_FILE_FORMAT:	MF-TYRE, MF-SWIFT, USER

OSRate

Specifies the oversampling rate of the tire model. With the value=5 the model is called five times in one simulation cycle with dt=1/5ms.

The next four parameters are used to specify the operation mode of the MF-Tyre/MF-Swift model. If operation modes of the MF-Swift model are used, please make sure the license server is specified by the environment variable MADLIC_LICENSE_FILE=<portnumber>@<licenseserver> For more informations see the MF-Tyre/MF-Swift 6.2 Help Manual.

TyreSide

Specifies on which side of the vehicle the tire is mounted.

- "Automatic", CarMaker will set the right side (left/right) automatically.
- "Symmetric" will remove the asymmetric behavior.

ContactMethod

Specifies the method used to calculate the tire-road contact point.

- "Smooth road" is the default method used with CarMaker.
- "Smooth circular cross" should be used for motorcycle tires.
- "3D short wavelength road" is part of the MF-Swift model and should be used for roads with short obstacles (e.g. bumps) or road surfaces containing wavelength smaller than twice the tyre radius.

Dynamics

Specifies the dynamics mode used.

- "Steady state" doesn't include any dynamic tire model behavior.
- "Transient (linear)" mode incorporates tire relaxation, using an empirical model for the relaxation lengths.
- "Transient (non-linear)" uses a physical approach to determine the lag.
- "Rigid ring" is part of the MF-Swift model, the tire belt is assumed to behave like a rigid body. The mass and moments of inertia of the tyre belt will be subtracted from the total wheel mass and moments of inertia automatically by CarMaker.

SlipForces

Specifies the components of the force and moment vector, which should be used during the simulation.

- "Combined" is the default method used with CarMaker, components are used in combined mode.
- "Combined + Turnslip" is part of the MF-Swift model, all components are used in combined mode and a turn slip mode is switch on.

It is also possible to switch off parts of the calculation, this is normally not necessary.

- "None" will not evaluate Magic Formula and only calculate the load force.
- "Longitudinal" will only calculate longitudinal components.
- "Lateral" will only calculate lateral components.
- "Uncombined" will calculate all components in uncombined mode.

17.8 Tire Model "MF-Tyre/MF-Swift 7.2"

To use the tire model *MF-Tire/MF-Swift 7.2* by TASS International, a corresponding dynamic library will be loaded, which is included in the CarMaker installation (currently supported for Windows, XENO and ds1006). The path to the library can also be set in the *SimParameters* file, if another library should be used. For use of MF-Swift options a separate license from TASS international is required. For details please refer to section 'TASS MF-Tyre/MF-Swift 7.2'.

AdamsPropertyFile = *PropertyFile*

Link to an ADAMS® tire property file. Its path is relative to directory of the tire infofile.

Supported FILE_TYPE:	"tir"
Supported FILE_VERSION:	2 or 3
Supported FILE_FORMAT:	"ASCII"
Supported PROPERTY_FILE_FORMAT:	MF-TYRE, MF-SWIFT, USER

The next four parameters are used to specify the operation mode of the MF-Tyre/MF-Swift model. For more informations see the MF-Tyre/MF-Swift 7 User Manual.

TyreSide

Specifies on which side of the vehicle the tire is mounted.

- "Automatic", CarMaker will set the right side (left/right) automatically.
- "Symmetric" will remove the asymmetric behavior.

ContactMethod

Specifies the method used to calculate the tire-road contact point.

- "Smooth road" is the default method used with CarMaker.

Dynamics

Specifies the dynamics mode used.

- "Steady state" doesn't include any dynamic tire model behavior.
- "Transient (linear)" mode incorporates tire relaxation, using an empirical model for the relaxation lengths.
- "Transient (non-linear)" uses a physical approach to determine the lag.
- "Rigid ring" is part of the MF-Swift model, the tire belt is assumed to behave like a rigid body. The mass and moments of inertia of the tyre belt will be subtracted from the total wheel mass and moments of inertia automatically by CarMaker.

SlipForces

Specifies the components of the force and moment vector, which should be used during the simulation.

- "Combined" is the default method used with CarMaker, components are used in combined mode.
- "Combined + Turnslip" is part of the MF-Swift model, all components are used in combined mode and a turn slip mode is switch on.

It is also possible to switch off parts of the calculation, this is normally not necessary.

- "None" will not evaluate Magic Formula and only calculate the load force.
- "Longitudinal" will only calculate longitudinal components.
- "Lateral" will only calculate lateral components.
- "Uncombined" will calculate all components in uncombined mode.

17.9 Tire Model "FTire"

To use the tire model *FTire* provided by *cosin*, a corresponding dynamic library will be loaded. The *cosin* tire package has to be installed before using the interface. To use the FTire models with CarMaker, a separate license from *cosin* is required to run the FTire library.

Please note that no additional *SimParameter* keys like for other external tire model are required to dynamically link the FTire library. It is sufficient, that an FTire user environment exists on the host PC used.

The CarMaker tire Infofile should contain the following keys:

TirePropertyFile = ***PropertyFile***

Link to the tire property file (.tir). Its path is relative to directory of the tire Infofile.

The following parameters are used in the *cosin* environment and can be set in CarMaker, too, for user convenience. Please find a detailed description of the parameters in the FTire documentation:

Animation = ***value***

Optional parameter. Can be used to overwrite the FTire animation mode set in the tire property file. Possible values are:

- 0 online animation for tire off
- 1 online animation for tire on
- 2 online animation for tire as before
- 3 online animation for tire off, offline animation for tire on
- 4 online animation for tire on, offline animation for tire off

AdditionalOutput = ***value***

Optional parameter. Flag to activate additional plot output to the .mtl (ascii) or .mtb (binary) file from the FTire library. Possible values are:

- 0 off (default)
- 1 on (only standard plot signals, ascii)
- 2 on (more plot signals, ascii)
- 3 on (all available plot signals, ascii)
- 4 on (only standard plot signals, binary)
- 5 on (more plot signals, binary)
- 6 on (all available plot signals, binary)

AdditionalOutputNames: ***text***

Optional parameter. Can be used to register up to ten signals from FTire as CarMaker User Accessible Quantities. For the signal names and their meaning please refer to the FTire documentation.

Example

```
AdditionalOutputNames:
   air temperature
   air pressure
```

InitialTireTemp = ***value***

Optional parameter. Defines the initial tire temperature of the filling gas and by the same value the mean tread surface temperature. This is only effective if the thermal model is activated. If not set, the environment temperature is used.

RoadTemp = ***value***

Optional parameter. Defines the road surface temperature. If not set, the environment temperature is used.

TirePressure = ***value***

Optional parameter. Set to change the inflation pressure for the current environment temperature at cold start.

Chapter 18

Trailer Model

18.1 Overview

Like the vehicle model the trailer model is a MESA VERDE multi body model consisting of the vehicle body, 2 or 4 wheel carriers, 2 or 4 wheels and a additional load mass. This trailer model suits passenger towing vehicles and therefore only supports trailers with single or twin axle.

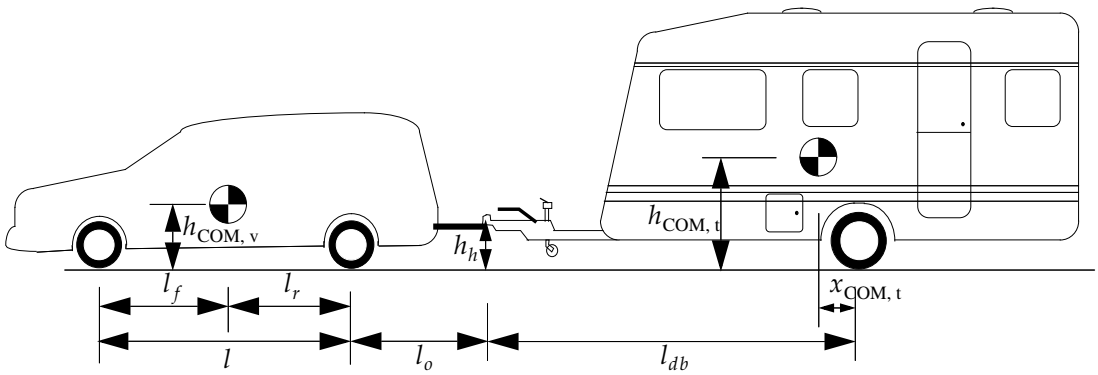


Figure 18.1: Common measures for a vehicle with trailer

Relevant quantities are:

$h_{COM,\,v}$	height of towing vehicles COM
$h_{COM,\,t}$	height of trailers COM
h_h	height of hitch
l	wheel base of towing vehicle
l_f	distance between COM and front axle
l_r	distance between COM and rear axle
l_o	hitch overhang
l_{db}	trailer wheel base
$x_{COM,\,t}$	distance between COM and axle

18.2 General Trailer Parameters

RefPointInputSystem = ***x y z***

This parameter specifies the origin of frame `Fr1` (in `FrD` coordinates). The coordinates XYZ point from the origin of frame `FrD` to the origin of frame `Fr1`. (see section 1.2 'CarMaker Axis Systems').

Movie.Skin.FName = ***FName***

Optional. Movie geometry object file name defines the geometry object of the trailer that is shown in the animation window.

Example `Movie.Skin.FName = Trailer.obj`

Vehicle.OuterSkin = ***rll.x rll.y rll.z fur.x fur.y fur.z***

Optional. The vehicle's outer skin box, defined by the points rear lower left (rll) and front upper right (fur).

Example `Vehicle.OuterSkin = -5.0 1.2 0.50.0 -1.2 2.5`

BasicDim.Length = ***lx ly lz***

Optional. The basic dimension length of the vehicle's outer skin box that is used if the parameter ***Vehicle.OuterSkin*** is not specified. The default values are calculated internally with the help of the hitch position, the body position and the track width.

Example `BasicDim.Length = 5.0 2.4 2.5`

ArticulationAngle_max = ***Angle***

Optional. This parameter specifies the maximum articulation angle ***Angle*** [deg] which is tolerated without issuing an error message. Higher angles lead to an abortion of the current TestRun. Default = 70.0.

nAxle = ***NumberOfAxles***

Number of trailer axles. Possible values: 1 or 2.

Vehicle.Model = *TrailerModel*

Specifies the trailer model. Possible models are *Semi_1Axle* and *Semi_2Axle*.

Tire.*<i>* = *TireParameterSet*

Specifies the tire parameter set to use for the front left (i=0), front right (i=1), rear left (i=2) and rear right (i=3) wheel. ***TireParameterSet*** is the name of a tire parameter set to be found in the directory `Data/Tire`.

Example

```
Tire.0 = DT_165_70R13
Tire.1 = DT_165_70R13
Tire.2 = DT_165_70R13
Tire.3 = DT_165_70R13
```

Virtual.Frc.pos = *x* *y* *z*

The virtual force/torque attacks the Trailer body in this defined position.

18.3 Mass Geometry

18.3.1 Overview

The representation of the distribution of mass in a material system is called mass geometry. Inertia properties are associated with the trailer-body, each of the two/four wheel carriers and each wheel. Additional body loads are modeled via additional masses and inertias as well. Three parameters have to be specified to define a mass element:

- The mass value of the body to define.
- The *center of mass* (`CoM`) of each body is defined. The center of mass is a geometrical point. The three scalar quantities are the position vector of `CoM` decomposed relative to the origin of the definition frame.
- The *inertia tensor* (= *second moments of mass = moments of inertia*) is a symmetric second-order tensor, which is specified by six scalar quantities `I = A B C D E F` (Frequently the off-diagonal elements (`D, E, F`) of the inertia tensor are neglected):

$$\begin{bmatrix} A & F & E \\ F & B & D \\ E & D & C \end{bmatrix} = \begin{bmatrix} \int_{\text{Body}} (y^2+z^2)dm & -\int_{\text{Body}} xydm & -\int_{\text{Body}} xzdm \\ \text{symmetry} & \int_{\text{Body}} (z^2+x^2)dm & -\int_{\text{Body}} yzdm \\ \text{symmetry} & \text{symmetry} & \int_{\text{Body}} (x^2+y^2)dm \end{bmatrix} \quad \textbf{(EQ 404)}$$

18.3.2 Parameters

Body.pos = ***x*** ***y*** ***z***
Body.mass = ***Mass***
Body.I = **...**

Trailer body with total mass *Body.mass* [kg].

Position of the center of mass (CoM) *Body.pos* is given in `FrD` (design configuration axis system). Inertia tensor *Body.I* at the CoM decomposed in the basis of the `FrD`.
The elements of the inertia tensor (section 7.1 'Overview')

TrimLoad.<i>.pos = ***x*** ***y*** ***z***
TrimLoad.<i>.mass = ***Mass***
TrimLoad.<i>.I = **...**

Bodies to trim mass contribution to that of a reference vehicle. Trim loads are fixed to vehicle body. <i> := 0, 1, 2, ...

Do not confuse TrimLoads with test run specific additional charges you may want to put on your vehicle.

WheelCarrier.<pos>.pos
WheelCarrier.<pos>.mass
WheelCarrier.<pos>.r

Wheel carrier at position <pos> (<pos>:=fl (front left), fr (front right), rl (rear left) and rr (rear right)). Represents all unsprung mass in the suspension (except the wheel).

Wheel.<pos>.pos
Wheel.<pos>.mass
Wheel.<pos>.r

Wheel at position <pos> (<pos>:=fl (front left), fr (front right), rl (rear left) and rr (rear right)). The wheel and all other rotating components (parts of the brake, ...).

18.3.3 Trailer with Twin Tires

Generally the trailer uses single tires for each wheel. In this case, the wheel carrier and the corresponding tire must be defined in the same position. In the same way like for vehicle, the trailer can be modelled with an axle using twin tires. Following parameters are used to activate the twin tires:

AxleF.TwinTiresOn = ***bool***
AxleR.TwinTiresOn = ***bool***

For details see section 7.3.3.

18.4 Suspension Kinematics and Compliance

The trailer suspension kinematics and compliance is defined in the same manner like for the vehicle suspension case except the trailer doesn't use the rack travel as a generalized coordinate (q_2). For details about the parameter description please refer to section 'Suspension Kinematics and Compliance'.

Besides the kinematics and compliance models for the vehicle there exist additional trailer kinematics models:

Suspension name	ModelKind	Description
Sleeve Axle	Sleeve	Kinematics of a sleeve axle
Crank Axle	Crank	Kinematics of a crank axle
Semi Trailing Arm	SemiTrailingArm	Kinematics of a semi trailing arm axle

For these three models the static camber and toe angle can be defined with the following parameters. As for the vehicle model the asterisk "*" is an abbreviation for the suspension, for kinematics or compliance, for the model number and the parameter side.

.Camber =** ***CamberAngle

Camber angle ***CamberAngle*** [rad] of left/right wheel in design configuration `FrD`. Positive sign means the distance on the upper side of the wheel is longer than on the bottom side.

.Toe =** ***ToeAngle

Toe angle ***ToeAngle*** [rad] of left/right wheel carrier in design configuration `FrD`. Positive sign means toe-in, negative sign means toe-out.

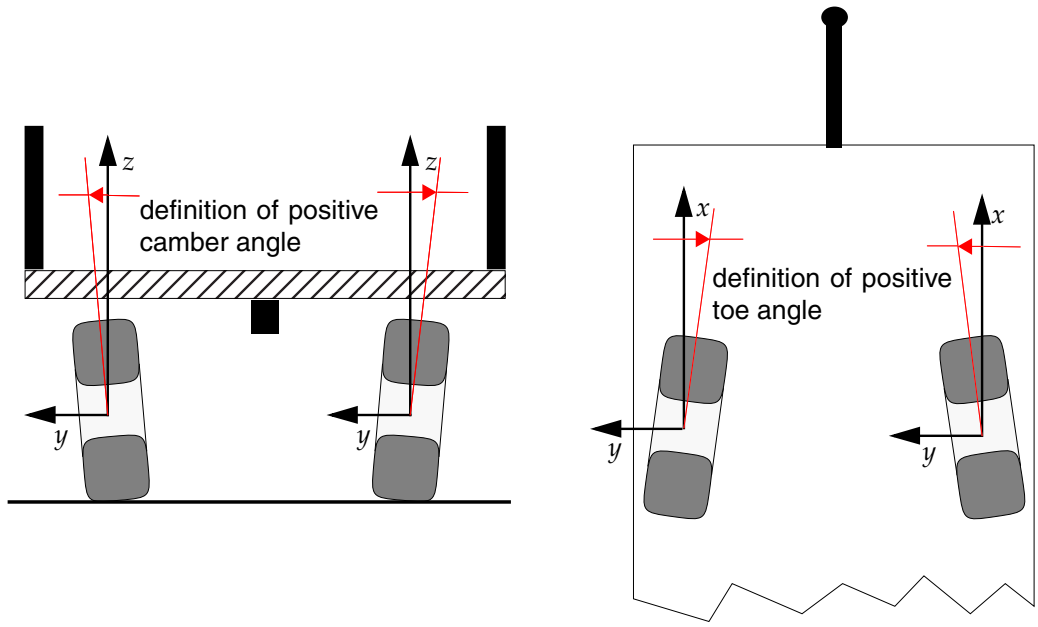


Figure 18.4: Sign conventions for camber- and toe-angle

18.4.1 Sleeve Axle

The simplest model of a suspension is the sleeve axle.

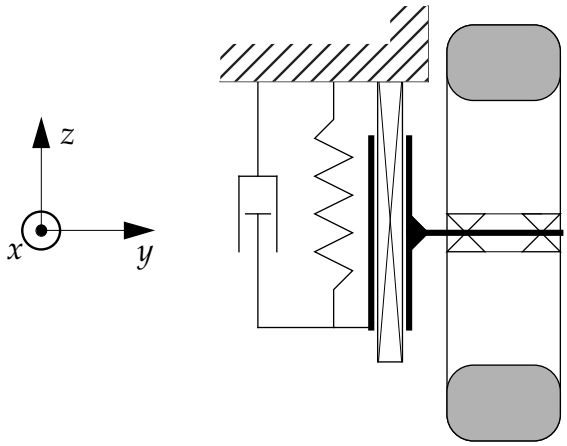


Figure 18.6: Model of a sleeve axle

It is characterized through a straight translation of the wheel carrier. In Figure 18.6 the translation in z coordinate direction is shown. There is no modification of toe or camber whilst compression.

18.4.2 Crank Axle

The most commonly used suspension for passenger cars trailers is the crank axle.

The crank axle 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 a orbit around the y axis of Figure 18.8.

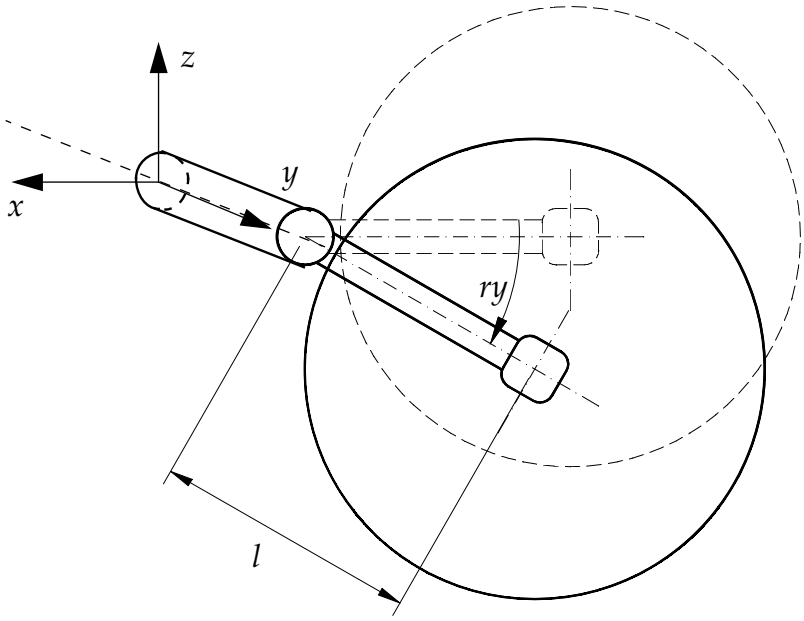


Figure 18.8: Principle of a crank axle

.lCrank =** ***length

Crank length [m].

18.4.3 Semi Trailing Arm Axle

The semi trailing arm axle uses the same principle as the crank axle. The difference is that the rotating axis is not along the y-axis like depicted in Figure 18.10. Through this different orientation the wheel practices a camber change.

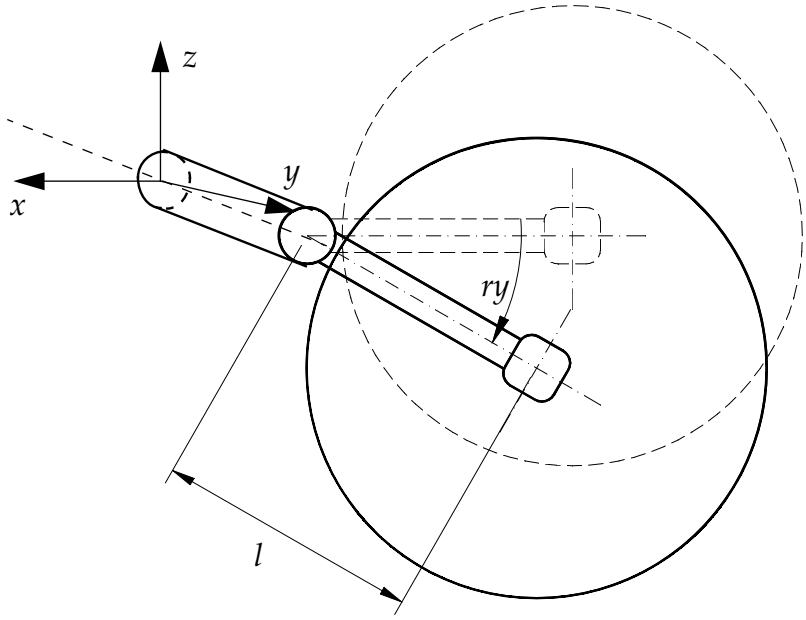


Figure 18.10: Principle of a semi trailing axle

Two points specify the rotation axle of the wheel carrier. The points must not be identical.

***.AxlePos1 =**	***x***	***y***	***z***

First point for wheel carrier rotation axle in `FrD` axis system.

***.AxlePos2 =**	***x***	***y***	***z***

Second point for wheel carrier rotation axle in `FrD` axis system.

18.4.4 Additional External Movement

In the same manner like for the vehicle the user has the possibility to apply additional movement of the wheel carrier besides the *Kinematics* and *Compliance*. Typically the additional movement is expressed with the translations tx, ty, tz and with the rotations rx, ry, rz of the wheel carrier. This movement is in the vehicle fixed frame Fr1.

The external movement can be manipulated via modification of correspond DVA quantities or directly using the parameters of the C code interface *TrSusp_Ext*, declared in `include/Car/Susp.h` header file.

Following table shows the user accessible quantities and the belonging parameter in the interface struct:

Table 18.1: External movement quantities and the belonging C-Variable

Quantity	C-Variable	Info
Tr.CFL.tx_ext	TrSusp_Ext.Kin_tExt[0][0]	External translation of the wheel carrier FL in direction x [m]
Tr.CFL.ty_ext	TrSusp_Ext.Kin_tExt[0][1]	External translation of the wheel carrier FL in direction y [m]
Tr.CFL.tz_ext	TrSusp_Ext.Kin_tExt[0][2]	External translation of the wheel carrier FL in direction z [m]
Tr.CFL.rx_ext	TrSusp_Ext.Kin_tExt[0][3]	External rotation of the wheel carrier FL in direction x [rad]
Tr.CFL.ry_ext	TrSusp_Ext.Kin_tExt[0][4]	External rotation of the wheel carrier FL in direction y [rad]
Tr.CFL.rz_ext	TrSusp_Ext.Kin_tExt[0][5]	External rotation of the wheel carrier FL in direction z [rad]
Tr.CFR.tx_ext	TrSusp_Ext.Kin_tExt[1][0]	External translation of the wheel carrier FR in direction x [m]
...	...	...
Tr.CRL.tx_ext	TrSusp_Ext.Kin_tExt[2][0]	External translation of the wheel carrier RL in direction x [m]
...	...	...
Tr.CRR.tx_ext	TrSusp_Ext.Kin_tExt[3][0]	External translation of the wheel carrier RR in direction x [m]
...	...	...

Remark

The velocity of the wheel carrier due to the external movement is not calculated in CarMaker. The user itself can apply additionally besides the translation also the velocity.

Table 18.2: shows the user accessible quantities for the external velocity of the wheel carrier and the corresponding C-Variable in the interface struct:

Table 18.2: External velocity quantities and the belonging C-Variable

Quantity	C-Variable	Info
Tr.CFL.txv_ext	TrSusp_Ext.Kin_vExt[0][0]	External trans. velocity of the wheel carrier FL in direction x [m/s]
...	...	...
Tr.CFL.rxv_ext	TrSusp_Ext.Kin_vExt[0][3]	External rot. velocity of the wheel carrier FL in direction x [rad/s]
...	...	...

18.5 Suspension Force Elements

The trailer uses the same suspension force element module like the vehicle model. The used parameters for this module are the same like for the vehicle. For details see section 'Suspension Force Elements'.

18.6 Hitch

18.6.1 Overview

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.

To ensure stabilization of the trailer different hitch improvements have been developed. The CarMaker trailer model supports the following hitch types:

- No stabilization, normal ball joint
- Four joint hitch
- Ball joint hitch with friction damper
- Ball joint with articulation angle dependent hydraulic damping
- Ball joint with torsion spring and rotational damper element

Parameters

Hitch.Kind = ***KindStr***

Specifies the type of hitch stabilization model to use. Possible values are:

HitchName	KindStr	Description
Ball	Ball	no stabilization, normal ball joint
Trapezoid	Trapez	four joint hitch
Ball with Friction	BallFric	ball joint hitch with friction damper
Ball with Damping	BallDamp	ball joint with articulation angle dependent hydraulic damping
Fifth Wheel	FifthWheel	ball joint with torsion spring and rotational damper element; used for truck trailer

Example `Hitch.Kind = BallFric`

Hitch.pos = ***x*** ***y*** ***z***

Center position of the trailers hitch. Coordinates are specified in trailers `FrD` axis system.

Hitch.c = ***dFrc/dq***

Hitch.c_x = ***dTrq/dq***
Hitch.c_y = ***dTrq/dq***
Hitch.c_z = ***dTrq/dq***

Optional. Overwrite the internally computed spring constant for trailer hitch.
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.

Hitch.k = ***dFrc/dqp***

Hitch.k_x = ***dFrc/dqp***
Hitch.k_y = ***dFrc/dqp***
Hitch.k_z = ***dFrc/dqp***

Optional. Overwrite the internally computed damper constant for trailer hitch.
Default: a default is computed internally by a ratio of spring constant to damper constant. Can be defined in all direction of space or in one certain direction.

18.6.2 Additional Parameters for Hitch “Ball”

No additional parameters needed.

18.6.3 Additional Parameters for Hitch “Trapezoid”

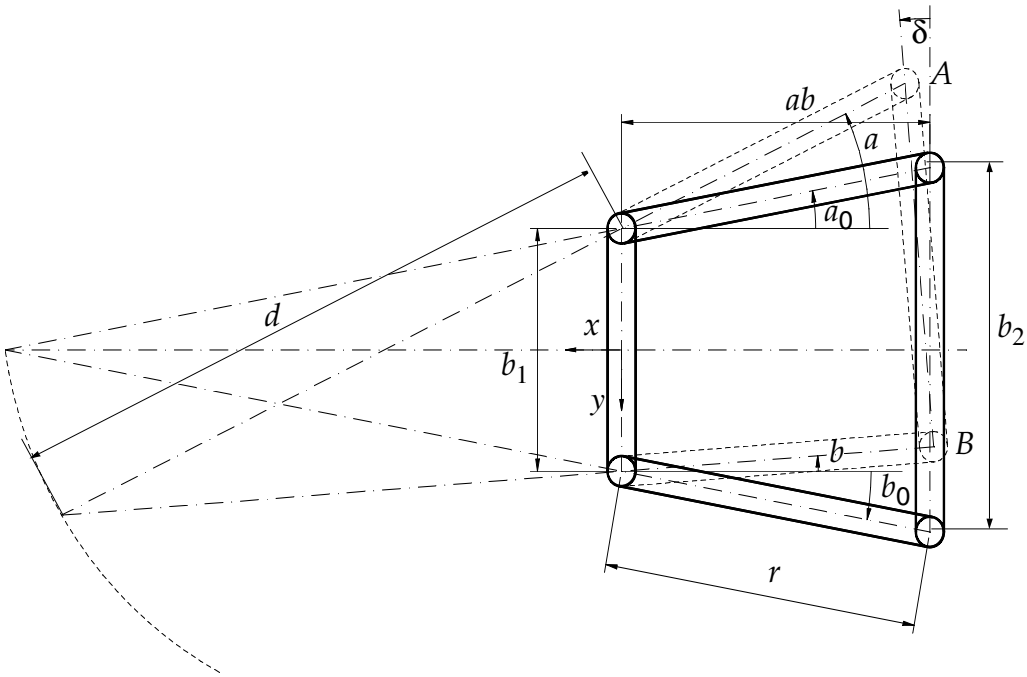


Figure 18.12: Trapetz hitch

Hitch.Trapez.ab = ***length***

Length of the trapezoid in direction of vehicles roll axle. See Figure 18.12.

Hitch.Trapez.b1 = ***length***

Short width of trapez. See Figure 18.12.

Hitch.Trapez.b2 = ***length***

Long width of trapez. See Figure 18.12.

18.6.4 Additional Parameters for Hitch “Ball with Friction”

Hitch.Fric.Adjust = ***factor***

Adaption parameter for calculation of stick-slip behavior. For this calculation trailer mass is altered by this factor. Use 1.0 as standard parameter.

Example `Hitch.Fric.Adjust = 1.0`

Hitch.Fric.myH = ***mu***

Static friction coefficient between ball-shaped hitch head and frictions elements.

Hitch.Fric.myG = ***mu***

Sliding friction coefficient between ball-shaped hitch head and frictions elements.

Hitch.Fric.FNorm = ***FrcNorm***

Normal force preloading the frictions elements against ball-shaped head.

Example `Hitch.Fric.FNorm = 2000`

18.6.5 Additional Parameters for Hitch “Ball with Damping”

Hitch.Damp.k = ***dFrc/dqp***

Hydraulic damper constant ***dFrc/dqp*** [Ns/rad] (equal in all directions of space).

Example `Hitch.Damp.k = 50`

18.6.6 Additional Parameters for Hitch "Fifth Wheel"

Hitch.FifthWheel.c = ***dTrq/dq***

Hitch.FifthWheel.c_x = ***dTrq/dq***
Hitch.FifthWheel.c_y = ***dTrq/dq***
Hitch.FifthWheel.c_z = ***dTrq/dq***

Torsion spring constant ***dTrq/dq*** [Nm/rad] can be defined in all direction of space or in one certain direction. Default: 0

Example

```
Hitch.FifthWheel.c = 1000
Hitch.FifthWheel.c_x = 100000
```

Hitch.FifthWheel.k = ***dTrq/dqp***

Hitch.FifthWheel.k_x = ***dTrq/dqp***
Hitch.FifthWheel.k_y = ***dTrq/dqp***
Hitch.FifthWheel.k_z = ***dTrq/dqp***

Rotational damper constant ***dTrq/dqp*** [Nms/rad] can be defined in all direction of space or in one certain direction. Default: 0

Example

```
Hitch.FifthWheel.k = 50
Hitch.FifthWheel.k_x = 7000
```

Hitch.FifthWheel.RotPlay = x y z

Specifies the rotational play in the 'FifthWheel' hitch for all directions. The values correspond to the minimal and maximal play. Unit: [deg]. Default: 0.0 0.0 0.0.

18.7 Brake

The brake module of the trailer has the same structure and interface as the brake module of the vehicle. For details of the structure and interface please refer to section 'Brake'.

Brake.Kind = *KindStr VersionId*

Selection of brake model to use. Following models are provided:

ModelName	KindStr	Description
""		Trailer without brake system
Overrun Brake	Overrun	Overrun brake, brake torque proportional to drawbar force
Overrun Brake with Friction	OverrunFric	Overrun brake, brake torque based on friction
Hydraulic	Hyd	Hydraulic model, consisting of hydraulic control and brake system

18.7.1 Brake model "Overrun"

This brake model consists of an overrun device, a transmission device and the wheel brakes.

Brake.Ratio

Ratio of the sum of all brake forces on all wheels to the force of the trailer hitch.
In EWG directive 71/320 "brake systems" a ratio of 5 is compulsory.

Example `Brake.Ratio = 5`

Brake.Fmin = *value*

Operating threshold of the overrun brake. Dimension: N

Example `Brake.Fmin = 200`

Brake.Delay = *value*

Time delay for brake force rise. A first order delay element is used. Too small values may lead to instabilities, too high values are not realistic. Dimension: s.

Example `Brake.Delay = 0.4`

18.7.2 Brake model “Overrun with Friction”

This brake model consists of an overrun device, a transmission device and the wheel brakes based on friction force.

Brake.Ratio

Ratio of the sum of all brake forces on all wheels to the force of the trailer hitch.
In EWG directive 71/320 “brake systems” a ratio of 5 is compulsory.

Example `Brake.Ratio = 5`

Brake.Fmin = ***value***

Operating threshold of the overrun brake. Dimension: N

Example `Brake.Fmin = 200`

Brake.Fric.Adjust = ***value***

Adaption parameter for calculation of stick-slip behavior. For this calculation trailer mass is altered by this factor. Use 1.0 as standard parameter.

Brake.Fric.myH = ***value***

Static friction coefficient between brake pads and brake drum (brake disk).

Example `Brake.Fric.myH = 0.55`

Brake.Fric.myG = ***value***

Sliding friction coefficient between brake pads and brake drum (brake disk).

Example `Brake.Fric.myG = 0.45`

18.7.3 Brake model “Hydraulic”

The trailer brake model *Hydraulic* works in the same manner as the hydraulic model in the vehicle, composed of a brake control and a brake system part. For detailed description of this model please refer to section 16.3 ’Brake model "Hydraulic"’.

For the brake control the same models can be used as for the vehicle (please refer to section 16.3.1 ’Hydraulic Brake Control’).

For the brake system following models are supported:

Brake.System.Kind = *KindStr VersionId*

ModelName	KindStr	Description
Pressure Distribution	TrPresDistrib	Brake torque proportional to driver brake pedal

Hydraulic Brake System model "Pressure Distribution"

The trailer brake model *TrPresDistrib* acts in similar manner like the vehicle brake model *PresDistrib*. This model don’t use an overrun brake. The braking torque is calculated as function of the brake pressure and the brake pressure is proportional to the driver brake pedal. For details about the model *PresDistrib* see section 16.4 ’Hydraulic Brake System "Pressure Distribution"’.

Brake.System.Pedal2pMC = *value*

Ratio driver brake pedal to braking pressure [bar].

Example
```
Brake.System.Pedal2pMC = 100
```

Brake.System.pMC2Trq = *Table*

Ratio braking pressure to brake torque [Nm/bar]. Specifies for each wheel a value.

Example
```
Trailer-1Axle:
Brake.System.pMC2Trq = 5   5
```

Example
```
Trailer-2Axle:
Brake.System.pMC2Trq = 10   10   10   10
```

Example
```
Trailer-3Axle:
Brake.System.pMC2Trq = 10    10    10    10    8    8
```

18.8 Aerodynamics

The trailer uses the same aerodynamic model as the towing vehicle. For general description see section 13.3.1 '"1D Look-Up Table"'.

18.9 User Accessible Quantities for Trailer

Please refer to section 24.15 'Trailer'.

Chapter 19

Power Flow Calculation

19.1 Introduction

CarMaker gives the possibility to analyze in detail the power flow while simulation. In the process, the applied power (e.g. the engine power) and all different losses can be observed. In this way, the contribution of each different kind of loss (powertrain, aerodynamics, ...) to the energy consumption can be investigated and optimized.

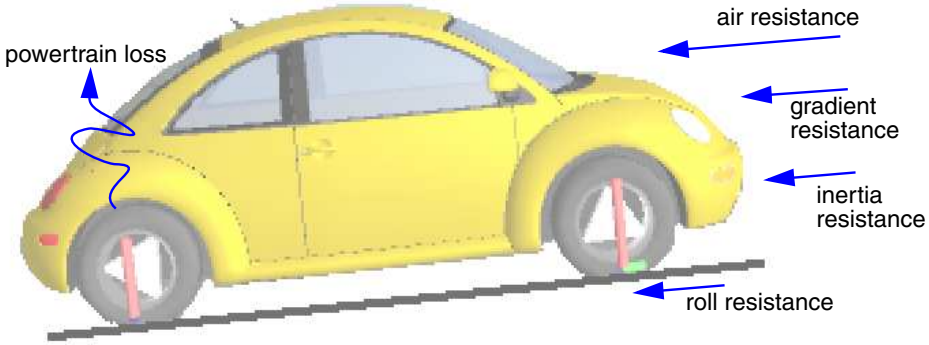


Figure 19.1: Different kinds of loss

19.2 Definition

The calculated quantities with PowerDelta describes the difference of each module between the input and output power. The correspond variables are in the global header file “PowerFlow.h” in the struct *PowerDelta*. The values are positive if there is a power loss (Power.Out < Power.In) and negative if there is a power gain (Power.Out > Power.In). At this point it isn’t known if the PowerDelta has been stored or has been irrecoverably lost.
To describe the stored and the lost power of each module, variables in the structs *PowerStore* and *PowerLoss* are defined.

19.3 Parameters

For activation of Power Flow calculation following parameter in *SimParameters-File* is used:

SimParam.PowerCalcOn = *bool*

Optional. Default is 1.

19.4 PowerDelta

19.4.1 Overview

Following PowerDeltas are calculated with CarMaker:

PowerDelta kinds	Detailed PowerDelta
PowerTrain	Engine, ISG, Motor
	Clutch
	GearBox, GearBoxM (motor gearbox), PlanetGear
	Differential (Front, Rear, Center)
	HangOn Clutch
	Inertia (DriveLine), Inertia (components before DriveLine)
	Flexible shafts
	Battery_LV, Battery_HV, PowerSupply
Tire	Longitudinal Slip
	Lateral Slip
	Vertical Deflection
	Camber Deflection
	Rolling Resistance
	Toe Slip
Suspension	Spring
	Damper
	Buffer
	Stabilizer
Aerodynamic	Drag Force, Side Force, Lift Force
	Roll Torque, Pitch Torque, Yaw Torque
Trailer	Hitch Forces and Torques
Gravitation	Road Inclination
Chassis	Inertia of vehicle body
Brake	Brake Torques

19.4.2 PowerTrain Delta

Engine

The PowerDelta in the engine *PowerDelta.PT.Engine* is calculated as follows:

$$PwrD_{Engine} = -Trq_{Engine} \cdot \omega_{Engine} \quad \text{(EQ 405)}$$

Variable	Description	UAQ
Trq_{Engine}	Torque of the engine	PT.Engine.Trq
ω_{Engine}	Rotational velocity of the engine	PT.Engine.rotv

ISG

The PowerDelta in the starter generator *PowerDelta.PT.ISG* is calculated as follows:

$$PwrD_{ISG} = PwrElec_{ISG} - Trq_{ISG} \cdot \omega_{ISG} \quad \text{(EQ 406)}$$

Variable	Description	UAQ
Trq_{ISG}	Torque of the starter generator ISG	PT.MotorISG.Trq
ω_{ISG}	Rotational velocity of starter generator	PT.MotorISG.rotv
$PwrElec_{ISG}$	Electric power of the starter generator	PT.MotorISG.PwrElec

Motor

The PowerDelta in the electric motor *PowerDelta.PT.Motor[iM]* is calculated as follows:

$$PwrD_{Motor} = PwrElec_{Motor} - Trq_{Motor} \cdot \omega_{Motor} \quad \text{(EQ 407)}$$

Variable	Description	UAQ
Trq_{Motor}	Torque of the electric motor	PT.Motor.Trq PT.Motor<iM>.Trq
ω_{Motor}	Rotational velocity of the electric motor	PT.Motor.rotv PT.Motor<iM>.rotv
$PwrElec_{Motor}$	Electric power of the electric motor	PT.Motor.PwrElec PT.Motor<iM>.PwrElec

Clutch

The PowerDelta in the clutch *PowerDelta.PT.Clutch* is calculated as follows:

$$PwrD_{Clutch} = Trq_{Eng2Clutch} \cdot \omega_{ClutchIn} - Trq_{Clutch2GearBox} \cdot \omega_{ClutchOut} \quad \text{(EQ 408)}$$

Variable	Description	UAQ
$Trq_{Eng2Clutch}$	Torque between engine and clutch	PT.Clutch.Trq_in
$Trq_{Clutch2GearBox}$	Torque between clutch and gearbox	PT.Clutch.Trq_out
$\omega_{ClutchIn}$	Rot. velocity of the clutch input shaft	PT.Clutch.rotv_in
$\omega_{ClutchOut}$	Rot. velocity of the clutch output shaft	PT.Clutch.rotv_out

GearBox

The PowerDelta in the gearbox *PowerDelta.PT.GearBox* is calculated as follows:

$$PwrD_{GearBox} = Trq_{Clutch2GearBox} \cdot \omega_{GearBoxIn} - Trq_{GearBox2DriveLine} \cdot \omega_{GearBoxOut} \quad \textbf{(EQ 409)}$$

Variable	Description	UAQ
$Trq_{Clutch2GearBox}$	Torque between clutch and gearbox	PT.GearBox.Trq_in
$Trq_{GearBox2DriveLine}$	Torque from gearbox to the driveline	PT.GearBox.Trq_out
$\omega_{GearBoxIn}$	Rot. velocity of the gearbox input shaft	PT.GearBox.rotv_in
$\omega_{GearBoxOut}$	Rot. velocity of the gearbox output shaft	PT.GearBox.rotv_out

In this gearbox PowerDelta is no gearbox inertia considered (only generalized torques). The gearbox inertia is considered in *PowerDelta.PT.Inert_DL* and in *PowerDelta.PT.Inert.*

GearBoxM (electric motor gearbox)

The PowerDelta in the electric motor gearbox *PowerDelta.PT.GearBoxM[iM]* is calculated as follows:

$$PwrD_{GearBoxM} = Trq_{GearBoxMIn} \cdot \omega_{GearBoxMIn} - Trq_{GearBoxMOut} \cdot \omega_{GearBoxMOut} \quad \textbf{(EQ 410)}$$

Variable	Description	UAQ
$Trq_{GearBoxMIn}$	Input torque to the electric motor gearbox	PT.GearBoxM.Trq_in PT.GearBoxM<iM>.Trq_in
$Trq_{GearBoxMOut}$	Output torque from the electric motor gearbox	PT.GearBoxM.Trq_out PT.GearBox<iM>.Trq_out
$\omega_{GearBoxMIn}$	Input rot. velocity of the electric motor gearbox input shaft	PT.GearBoxM.rotv_in PT.GearBoxM<iM>.rotv_in
$\omega_{GearBoxMOut}$	Output rot. velocity of the electric motor gearbox output shaft	PT.GearBoxM.rotv_out PT.GearBoxM<iM>.rotv_out

In this gearbox PowerDelta is no gearbox inertia considered (only generalized torques). The gearbox inertia is considered in *PowerDelta.PT.Inert_DL* and in *PowerDelta.PT.Inert.*

PlanetGear

The PowerDelta in the planetgear *PowerDelta.PT.PlanetGear* is only calculated using the hybrid powertrain "PowerSplit". Without describing in detail, the simplified calculation of the PowerDelta is:

$$PwrD_{PG} = Trq_{Cage} \cdot \omega_{Cage} - Trq_{Sun} \cdot \omega_{Sun} - Trq_{Ring} \cdot \omega_{Ring} \quad \textbf{(EQ 411)}$$

Variable	Description	UAQ
Trq_{Cage}	Input torque to the cage gear	
Trq_{Sun}	Output torque at the sun gear	
Trq_{Ring}	Output torque at the ring gear	
ω_{Cage}	Rot. velocity of the cage gear shaft	PT.Engine.rotv
ω_{Sun}	Rot. velocity of the sun gear shaft	PT.MotorISG.rotv
ω_{Ring}	Rot. velocity of the ring gear shaft	PT.Motor.rotv

Differential

The PowerDelta in the differential *PowerDelta.PT.FDiff* is calculated as follows:

$$PwrD_{FDiff} = Trq_{FDiffIn} \cdot \omega_{FDiffIn} - (Trq_{DriveFL} \cdot \omega_{WheelFL} + Trq_{DriveFR} \cdot \omega_{WheelFR}) \quad \textbf{(EQ 412)}$$

The PowerDelta in the differential *PowerDelta.PT.RDiff* is calculated as follows:

$$PwrD_{RDiff} = Trq_{RDiffIn} \cdot \omega_{RDiffIn} - (Trq_{DriveRL} \cdot \omega_{WheelRL} + Trq_{DriveRR} \cdot \omega_{WheelRR}) \quad \textbf{(EQ 413)}$$

The PowerDelta in the differential *PowerDelta.PT.CDiff* is calculated as follows:

$$PwrD_{CDiff} = Trq_{CDiffIn} \cdot \omega_{CDiffIn} - (Trq_{FDiffIn} \cdot \omega_{FDiffIn} + Trq_{RDiffIn} \cdot \omega_{RDiffIn}) \quad \textbf{(EQ 414)}$$

Variable	Description	UAQ
$Trq_{FDiffIn}$ $Trq_{RDiffIn}$ $Trq_{CDiffIn}$	Input torque to the front/rear/central differential	PT.Gen.DL.FDiff.Trq_Input PT.Gen.DL.RDiff.Trq_InputPT .Gen.DL.CDiff.Trq_Input
$Trq_{Drive[i]}$	Drive torque at the i wheel (i=<pos> = FL, FR, RL, RR)	PT.W<pos>.TrqDrive
$\omega_{FDiffIn}$, $\omega_{RDiffIn}$ $\omega_{CDiffIn}$	Rotational velocity of the front/rear/central differential input shaft	PT.Gen.DL.FDiff.rotv_in PT.Gen.DL.RDiff.rotv_in PT.Gen.DL.CDiff.rotv_in
$\omega_{Wheel[i]}$	Rotational velocity of the i wheel	PT.W<pos>.rotv

In the differential PowerDelta is no differential inertia considered (only generalized torques). The differential inertia is considered in *PowerDelta.PT.Inert_DL.*

HangOn-Clutch

For the driveline configuration “Gen2p2Front“ PowerDelta in the hang-on clutch *PowerDelta.PT.HangOn* is calculated as follows:

$$PwrD_{HangOn} = Trq_{HangOn} \cdot \omega_{FDiffIn} - Trq_{HangOn} \cdot \omega_{RDiffIn} \quad \textbf{(EQ 415)}$$

Variable	Description	UAQ
Trq_{HangOn}	Torque in the hangon-clutch	PT.Gen.DL.HangOn.Trq_Cpl2B
$\omega_{FDiffIn}$	Rotational velocity of the front differential input shaft	PT.Gen.DL.FDiff.rotv_in
$\omega_{RDiffIn}$	Rotational velocity of the rear differential input shaft	PT.Gen.DL.RDiff.rotv_in

For the driveline configuration “Gen2p2Rear“ PowerDelta in the hang-on clutch is calculated in the same way besides the expression is inverse.

Inertia

In *PowerDelta.PT.Inert* are all powertrain components like engine, starter generator, electric motor, clutch and input gearbox inertias considered. As representative the calculation of the PowerDelta for engine inertia is shown:

$$PwrD_{InertEng} = I_{Engine} \cdot \dot{\omega}_{Engine} \cdot \omega_{Engine} \quad \text{(EQ 416)}$$

Variable	Description	UAQ
I_{Engine}	Engine inertia incl. clutch input shaft inertia	
$\dot{\omega}_{Engine}$	Rot. acceleration of the engine shaft	
ω_{Engine}	Rotational velocity of the engine shaft	PT.Engine.rotv

DriveLine Inertia

The PowerDelta due to the driveline inertia *PowerDelta.PT.Inert_DL* is calculated as follows:

$$PwrD_{InertDL} = \sum_{i = FL, FR, RL, RR} I_{Red[i]} \cdot \dot{\omega}_{Wheel[i]} \cdot \omega_{Wheel[i]} \quad \text{(EQ 417)}$$

Variable	Description	UAQ
$I_{Red[i]}$	Wheel inertia incl. the gearbox and the differential inertia	
$\dot{\omega}_{Wheel[i]}$	Rotational acceleration of the wheel i	
$\omega_{Wheel[i]}$	Rotational velocity of the wheel i	PT.W<pos>.rotv

Flexible Shafts

The PowerDelta due to the total shaft torque *PowerDelta.PT.Shafts* is calculated as follows:

$$PwrD_{Shafts} = \sum \Delta\omega_{Shaft} \cdot Trq_{Shaft} = \sum \Delta\omega_{Shaft} \cdot (\Delta\omega_{Shaft} \cdot k + \Delta\varphi_{Shaft} \cdot c) \quad \text{(EQ 418)}$$

The PowerDelta due to the stored shaft spring torque *PowerDelta.PT.Spring_DL* is calculated as follows:

$$PwrD_{SpringDL} = \sum \Delta\omega_{Shaft} \cdot Trq_{ShaftSpring} = \sum \Delta\omega_{Shaft} \cdot (\Delta\varphi_{Shaft} \cdot c) \quad \text{(EQ 419)}$$

Variable	Description	UAQ
c , k	Shaft rotational stiffness, damping	
$\Delta\omega_{Shaft}$	Rotational speed difference in the shaft	
$\Delta\varphi_{Shaft}$	Rotational angle difference in the shaft	

Battery LV, Battery HV

The PowerDelta in the low voltage battery *PowerDelta.PT.Battery_LV* and in high voltage battery *PowerDelta.PT.Battery_HV* is calculated as follows:

$$PwrD_{Battery} = -I_{Battery} \cdot U_{Battery} \quad \textbf{(EQ 420)}$$

Variable	Description	UAQ
$I_{Battery}$	Battery current	PT.BattLV.Current PT.BattHV.Current
$U_{Battery}$	Battery voltage	PT.PwrSupply.LV.Voltage PT.PwrSupply.HV1.Voltage

PowerSupply

The PowerDelta in *PowerDelta.PT.PowerSupply* considers the power flow in the DC/DC converter between the low and high voltage level.

Total of PowerTrain

The total PowerDelta of the powertrain *PowerDelta.PT.Total* is the sum of all powertrain parts.

19.4.3 Brake Delta

The PowerDelta in the brake *i* (*FL,FR,RL,RR*) *PowerDelta.Brake[i].Total* is calculated as follows:

$$PwrD_{Brake[i]} = -Trq_{Brake2Wheel[i]} \cdot \omega_{Wheel[i]} \quad \textbf{(EQ 421)}$$

Variable	Description	UAQ
$Trq_{Brake2Wheel[i]}$	Torque acting from brake to wheel i	PT.W<pos>.Trq_B2W
$\omega_{Wheel[i]}$	Rotational velocity of the wheel i	PT.W<pos>.rotv

Total of Brake

The total PowerDelta of the brake *PowerDelta.Brake_Total* is the sum of all brake parts.

19.4.4 Tire Delta

Longitudinal Slip

The PowerDelta due to longitudinal slip *PowerDelta.Tire[i].LongSlip* is calculated as follows:

$$PwrD_{LongSlip[i]} = Frc_x^W(P_{[i]}) \cdot v_{xRel}^W(P_{[i]}) = Frc_x^W(P_{[i]}) \cdot (\omega_{Wheel[i]} \cdot R_{eff[i]} - v_x^W(P_{[i]})) \quad \textbf{(EQ 422)}$$

Variable	Description	UAQ
$Frc_x^W(P_{[i]})$	Longitudinal force in tire contact point P of tire i, expressed in frame FrW	Car.Fx<pos>
$\omega_{Wheel[i]}$	Rotational velocity of the wheel i	both multiplied together are Car.v<pos>
$R_{eff[i]}$	Effective (kinematic) tire radius of tire i	

Variable	Description	UAQ
$v_x^W(P_{[i]})$	Longitudinal velocity of tire contact point P, expressed in tire fixed frame FrW	Car.C<pos>.P.v01_W.x

Lateral Slip

The PowerDelta due to lateral slip *PowerDelta.Tire[i].LatSlip* is calculated as follows:

$$PwrD_{LatSlip[i]} = Frc_y^W(P_{[i]}) \cdot v_{yRel}^W(P_{[i]}) = -Frc_y^W(P_{[i]}) \cdot v_y^W(P_{[i]}) \quad \textbf{(EQ 423)}$$

Variable	Description	UAQ
$Frc_y^W(P_{[i]})$	Lateral force in tire contact point P of tire i, expressed in frame FrW	Car.Fy<pos>
$v_y^W(P_{[i]})$	Lateral relative velocity of tire contact point P, expressed in tire fixed frame FrW	Car.C<pos>.P.v01_W.y

Vertical Deflection

The PowerDelta due to vertical deflection *PowerDelta.Tire[i].VertDefl* is calculated as follows:

$$PwrD_{VertDefl[i]} = Frc_z^W(P_{[i]}) \cdot v_{zRel}^W(C_{[i]}) \quad \textbf{(EQ 424)}$$

Variable	Description	UAQ
$Frc_z^W(P_{[i]})$	Normal force in tire contact point P of tire i, expressed in frame FrW	Car.Fz<pos>
$v_{zRel}^W(C_{[i]})$	Relative Velocity of the wheel carrier center point C for the wheel i, expressed in frame FrW	

Camber Deflection

The PowerDelta due to camber deflection *PowerDelta.Tire[i].CambDefl* is calculated as follows:

$$PwrD_{CambDefl[i]} = -Trq_x^W(P_{[i]}) \cdot \omega_x^W(C_{[i]}) \quad \textbf{(EQ 425)}$$

Variable	Description	UAQ
$Trq_x^W(P_{[i]})$	Overturning torque in tire contact point P of tire i, expressed in frame FrW	Car.TrqOvert<pos>
$\omega_x^W(C_{[i]})$	Rotational velocity of wheel carrier i in tire center point C, expressed in frame FrW	

Rolling Resistance

The PowerDelta due to rolling resistance *PowerDelta.Tire[i].RollResist* is calculated as follows:

$$PwrD_{RollResist[i]} = -Trq_y^W(P_{[i]}) \cdot \omega_{Wheel[i]} \quad \text{(EQ 426)}$$

Variable	Description	UAQ
$Trq_y^W(P_{[i]})$	Rolling resistance torque in tire i contact point P, expressed in frame FrW	Car.TrqRoll<pos>
$\omega_{Wheel[i]}$	Rotational velocity of the wheel i	PT.W<pos>.rotv

Toe Slip

The PowerDelta due to toe slip *PowerDelta.Tire[i].ToeSlip* is calculated as follows:

$$PwrD_{ToeSlip[i]} = -Trq_z^W(P_{[i]}) \cdot \omega_z^W(C_{[i]}) \quad \text{(EQ 427)}$$

Variable	Description	UAQ
$Trq_z^W(P_{[i]})$	Aligning torque in tire contact point P of tire i, expressed in frame FrW	Car.TrqAlign<pos>
$\omega_z^W(C_{[i]})$	Rotational velocity of wheel carrier i in tire center point C, expressed in frame FrW	

Total of Tire

The total PowerDelta of tire *i PowerDelta.Tire[i].Total* is the sum of all PowerDelta parts for the tire *i*.

The total tire PowerDelta of vehicle *PowerDelta.Tire_Total* is the sum of all tire PowerDeltas *PowerDelta.Tire[i].Total.*

19.4.5 Suspension Delta

Spring

The PowerDelta in the spring *PowerDelta.Susp[i].Spring* is calculated as follows:

$$PwrD_{Spring[i]} = -Frc_{Spring[i]} \cdot v_{Spring[i]} \quad \text{(EQ 428)}$$

Variable	Description	UAQ
$Frc_{Spring[i]}$	Spring force in suspension i	Car.Spring<pos>.Frc
$v_{Spring[i]}$	Velocity of the spring i	

There is no suspension PowerDelta calculation due to external forces.

Damper

The PowerDelta in the damper *PowerDelta.Susp[i].Damper* is calculated as follows:

$$PwrD_{Damper[i]} = -Frc_{Damper[i]} \cdot v_{Damper[i]} \quad \text{(EQ 429)}$$

Variable	Description	UAQ
$Frc_{Damper[i]}$	Damper force in suspension i	Car.Damp<pos>.Frc
$v_{Damper[i]}$	Velocity of the damper i	Car.Damp<pos>.v

Buffer

The PowerDelta in the buffer *PowerDelta.Susp[i].Buffer* is calculated as follows:

$$PwrD_{Buffer[i]} = -Frc_{Buffer[i]} \cdot v_{Buffer[i]} \quad \text{(EQ 430)}$$

Variable	Description	UAQ
$Frc_{Buffer[i]}$	Buffer force in suspension i	Car.Buffer<pos>.Frc
$v_{Buffer[i]}$	Velocity of the buffer i	

Stabilizer

The PowerDelta in the stabilizer *PowerDelta.Susp[i].Stabi* is calculated as follows:

$$PwrD_{Stabi[i]} = -Frc_{Stabi[i]} \cdot v_{Stabi[i]} \quad \text{(EQ 431)}$$

Variable	Description	UAQ
$Frc_{Stabi[i]}$	Stabilizer force in suspension i	Car.Stabi<pos>.Frc
$v_{Stabi[i]}$	Velocity of the stabilizer i	

Total of Suspension

The total PowerDelta of suspension *i PowerDelta.Susp[i].Total* is the sum of all PowerDelta parts for the suspension *i*.

The total suspension PowerDelta of vehicle *PowerDelta.Susp_Total* is the sum of all suspension PowerDeltas *PowerDelta.Susp[i].Total.*

19.4.6 Aerodynamic Delta

Drag Force

The PowerDelta due to aerodynamic drag force *PowerDelta.Aero.Drag* is calculated as follows:

$$PwrD_{Drag} = Frc_x^1(A) \cdot v_{xRel}^1(A) \quad \text{(EQ 432)}$$

Variable	Description	UAQ
$Frc_x^1(A)$	Aerodynamic drag force in the point of attack A, expressed in frame Fr1	Car.Aero.Frc_1.x

Variable	Description	UAQ
$v^1_{xRel}(A)$	Relative velocity wind-vehicle in x-direction in point of attack A, expressed in frame Fr1	Car.Aero.vres_1.x

Side Force

The PowerDelta due to aerodynamic side force *PowerDelta.Aero.Side* is calculated as follows:

$$PwrD_{Side} = Frc^1_y(A) \cdot v^1_{yRel}(A) \quad \textbf{(EQ 433)}$$

Variable	Description	UAQ
$Frc^1_y(A)$	Aerodynamic side force in the point of attack A, expressed in frame Fr1	Car.Aero.Frc_1.y
$v^1_{yRel}(A)$	Relative velocity wind-vehicle in y-direction in point of attack A, expressed in frame Fr1	Car.Aero.vres_1.y

Lift Force

The PowerDelta due to aerodynamic lift force *PowerDelta.Aero.Lift* is calculated as follows:

$$PwrD_{Lift} = Frc^1_z(A) \cdot v^1_{zRel}(A) \quad \textbf{(EQ 434)}$$

Variable	Description	UAQ
$Frc^1_z(A)$	Aerodynamic lift force in the point of attack A, expressed in frame Fr1	Car.Aero.Frc_1.z
$v^1_{zRel}(A)$	Relative velocity wind-vehicle in z-direction in point of attack A, expressed in frame Fr1	Car.Aero.vres_1.z

Roll Torque

The PowerDelta due to aerodynamic roll moment *PowerDelta.Aero.Roll* is calculated as follows:

$$PwrD_{Roll} = Trq^1_x(A) \cdot \omega^1_x(A) \quad \textbf{(EQ 435)}$$

Variable	Description	UAQ
$Trq^1_x(A)$	Aerodynamic roll moment in the point of attack A, expressed in frame Fr1	Car.Aero.Trq_1.x
$\omega^1_x(A)$	Rotational velocity of vehicle in point of attack A, expressed in frame Fr1	

Pitch Torque

The PowerDelta due to aerodynamic pitch moment *PowerDelta.Aero.Pitch* is calculated as follows:

$$PwrD_{Pitch} = Trq_y^1(A) \cdot \omega_y^1(A) \quad \text{(EQ 436)}$$

Variable	Description	UAQ
$Trq_y^1(A)$	Aerodynamic pitch moment in the point of attack A, expressed in frame Fr1	Car.Aero.Trq_1.y
$\omega_y^1(A)$	Rotational velocity of vehicle in point of attack A, expressed in frame Fr1	

Yaw Torque

The PowerDelta due to aerodynamic yaw moment *PowerDelta.Aero.Yaw* is calculated as follows:

$$PwrD_{Yaw} = Trq_z^1(A) \cdot \omega_z^1(A) \quad \text{(EQ 437)}$$

Variable	Description	UAQ
$Trq_z^1(A)$	Aerodynamic yaw moment in the point of attack A, expressed in frame Fr1	Car.Aero.Trq_1.z
$\omega_z^1(A)$	Rotational velocity of vehicle in point of attack A, expressed in frame Fr1	

Total of Aerodynamic

The total PowerDelta of the aerodynamic *PowerDelta.Aero.Total* is the sum of all aerodynamic parts.

19.4.7 Trailer Delta

Hitch Force x,y,z

The PowerDelta due to hitch force in x-direction *PowerDelta.Hitch.Frc_x* is calculated as follows:

$$PwrD_{Frcx} = Frc_x^1(H) \cdot v_x^1(H) \quad \text{(EQ 438)}$$

Variable	Description	UAQ
$Frc_x^1(H)$	Hitch force in the hitch point H, expressed in frame Fr1	Tr.Hitch.Frc2Tr.x_1
$v_x^1(H)$	Vehicle velocity in the hitch point H, expressed in frame Fr1	

The calculation is in the same way for the hitch force y and the hitch force z.

Hitch Torque x, y, z

The PowerDelta due to hitch torque in x-direction *PowerDelta.Hitch.Trq_x* is calculated as follows:

$$PwrD_{Trqx} = Trq_x^1(H) \cdot \omega_x^1(H) \quad \text{(EQ 439)}$$

Variable	Description	UAQ
$Trq_x^1(H)$	Hitch torque in the hitch point H, expressed in frame Fr1	Tr.Hitch.Trq2Tr.x_1
$\omega_x^1(H)$	Vehicle rotational velocity in the hitch point H, expressed in frame Fr1	

Total of Hitch

The total PowerDelta for the trailer hitch *PowerDelta.Hitch.Total* is the sum of all hitch parts.

19.4.8 Gravitation Delta

The PowerDelta due to the inclination of the road *PowerDelta.Grvt* is calculated as follows:

$$PwrD_{Grvt} = M_{Vhcl} \cdot g \cdot v_z^0(G) \quad \text{(EQ 440)}$$

Variable	Description	UAQ
M_{Vhcl}	Total mass of the vehicle (incl. loads)	
g	Gravity acceleration (9.806m/s^2)	
$v_z^0(G)$	Vehicle velocity in z-direction in the center of mass G, expressed in frame Fr0	

19.4.9 Chassis Inertia Delta

The PowerDelta due to the vehicle inertia *PowerDelta.Inert_Chassis* is calculated as follows:

$$PwrD_{Chassis} = \sum_{i=x,y,z} \left(a_i^0(G) \cdot v_i^0(G) \cdot M_{Vhcl} + \omega_i^0 \cdot \sum_j (I_{ij} \cdot \dot{\omega}_j^0) \right) \quad \text{(EQ 441)}$$

Variable	Description	UAQ
M_{Vhcl}	Total mass of the vehicle (incl. loads)	
$a_i^0(G)$	Vehicle acceleration in i-direction in the center of mass G, expressed in frame Fr0	Car.Gen.a<i>
$v_i^0(G)$	Vehicle velocity in i-direction in the center of mass G, expressed in frame Fr0	Car.Gen.v<i>
I_{ij}	Inertia of vehicle	
ω_i^0	Rotational velocity of vehicle in i-direction, expressed in frame Fr0	
$\dot{\omega}_j^0$	Rotational acceleration of vehicle in j-direction, expressed in frame Fr0	

19.5 PowerStore

PowerTrain Store

The total stored power in the powertrain *PowerStore.PT* is the sum of all power due to the inertia of powertrain components, the shaft spring torque and the battery power:

$$PwrS_{PowerTrain} = PwrD_{Inert} + PwrD_{InertDL} + PwrD_{SpringDL} + PwrD_{Battery} \quad \textbf{(EQ 442)}$$

with $PwrD_{Battery} = PwrD_{BatteryLV} + PwrD_{BatteryHV}$

Brake Store

There is zero stored power due to the brake torques *PowerStore.Brake*.

Suspension Store

The total stored power in the suspensions *PowerStore.Susp* is the sum of following suspension PowerDeltas:

$$PwrS_{Suspension} = \sum_{i = FL, FR, RL, RR} (PwrD_{Spring[i]} + PwrD_{Buffer[i]} + PwrD_{Stabi[i]}) \quad \textbf{(EQ 443)}$$

Tire Store

There is zero stored power in the tires *PowerStore.Tire*.

Aerodynamic Store

There is zero stored power due to aerodynamic forces and torques *PowerStore.Aero*.

Hitch Store

Assuming of zero stored power due to trailer hitch forces and torques *PowerStore.Hitch*.

Gravitation Store

The total stored power due to road inclination *PowerStore.Grvt* is the total gravitation PowerDelta:

$$PwrS_{Grvt} = PwrD_{Grvt} \quad \textbf{(EQ 444)}$$

Chassis Inartia Store

The total stored power due to inertia of the vehicle body *PowerStore.Inert_Chassis* is the total chassis PowerDelta:

$$PwrS_{Chassis} = PwrD_{Chassis} \quad \textbf{(EQ 445)}$$

19.6 PowerLoss

For each module (PT, Brake, Suspension, Tire, ...) the lost power *PowerLoss.<Module>* is the difference of PowerDelta and PowerStore:

$$PwrL_{Module} = PwrD_{Module} - PwrS_{Module} \quad \text{(EQ 446)}$$

19.7 User Accessible Quantities

Please refer to section 24.13 'Power Flow Calculation'.

19.8 Manipulation of Power Flow Calculation

Each quantity of power flow analysis is a global variable, defined in the header file *PowerDelta.h*. The variables can be manipulated in the function *User_Calc()*, in the file *User.c*.

Example

```
int
User_Calc (double dt)
{
/* Manipulation of powertrain parts */
PowerDelta.PT.Engine += PowerEngine_Ext;
PowerDelta.PT.GearBox += PowerGearBox_Ext;

/* Total PowerDelta of PowerTrain */
PowerDelta.PT.Total =
+ PowerDelta.PT.Engine
   + PowerDelta.PT.Clutch + PowerDelta.PT.GearBox
+ PowerDelta.PT.FDiff + PowerDelta.PT.RDiff
   + PowerDelta.PT.CDiff + PowerDelta.PT.HangOn
   + PowerDelta.PT.Inert_Eng + PowerDelta.PT.Inert_DL;

/* Total PowerDelta */
PowerDelta.Total =
+ PowerDelta.PT.Total + PowerDelta.Tire_Total
+ PowerDelta.Susp_Total + PowerDelta.Brake_Total
   + PowerDelta.Aero.Total + PowerDelta.Inert_Chassis
   + PowerDelta.Hitch.Total + PowerDelta.Grvt;

/* Total PowerLoss of PowerTrain */
PowerLoss.PT =PowerDelta.PT.Total - PowerStore.PT;

/* Total PowerLoss */
PowerLoss.Total =
+ PowerLoss.PT + PowerLoss.Brake + PowerLoss.Tire
+ PowerLoss.Susp + PowerLoss.Aero + PowerLoss.Hitch
+ PowerLoss.Grvt + PowerLoss.Inert_Chassis;

return 0;
}
```

Chapter 20
Sensors

20.1 Introduction

Sensor models are required in many and varied use cases from developing functions on system level to developing the sensor component itself. The requirements for a suitable sensor model are as diverse as the use cases. The current chapter introduces a use case driven classification of sensor models. It might serve as guideline, which sensor model fits best to the respective needs.

20.1.1 Ideal Sensors for Rapid Prototyping/Function Development

Ideal Sensors are a good choice in an early stage of function development. They support the user in answering the question whether the function under development works in principle. Ideal Sensors extract the relevant information from the environment and deliver it in a form suitable to the function interface (e.g. object list). In case of imaging sensors tracking of objects and target selection is also included. Since Ideal Sensors are technology independent, they are simple to use with low effort for parametrization. Switching sensors can be easily realized (e.g. radar instead of ultrasonic for near range surveillance).

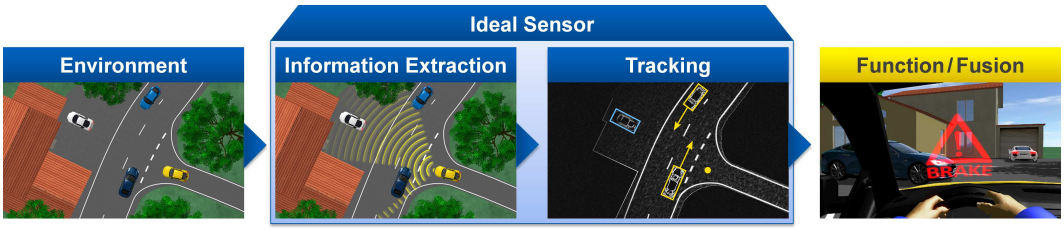


Figure 20.1: Ideal Sensor for rapid prototyping/function development (blue: CM, yellow: user)

Ideal Sensors also provide a suitable interface for user defined models for function development. The extracted information can be easily enhanced by user specific calculations to get a custom physical sensor model. For instance, the Free Space Sensor can be extended with an intensity based detection to model a lidar.

20.1.2 High Fidelity Sensors for Function Development/Testing

High Fidelity (HiFi) Sensors are designed for function development and testing. They allow for answering the question whether the function under development works in general. HiFi Sensors consider basic technology specific effects caused by propagation of the respective signal in the environment as well as by processing algorithms. They also provide the information in a form appropriate to the function interface (e.g. object list).

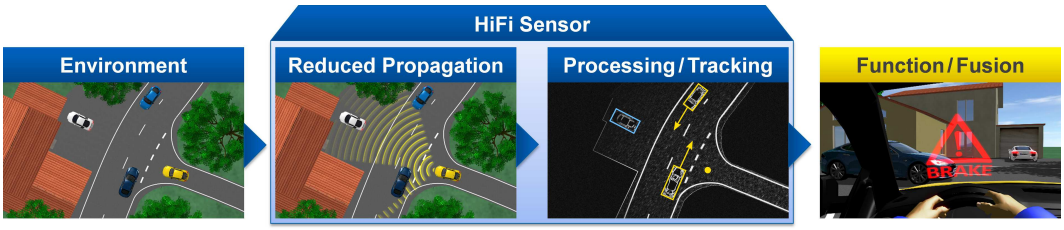


Figure 20.2: HiFi Sensor for function development and testing (blue: CM, yellow: user)

20.1.3 Raw Signal Interfaces for Component Development/Testing

Raw Signal Interfaces (RSIs) in contrast are made for component development and testing. They focus on detailed physical effects by propagation of the respective signal in the environment. RSIs do not offer a processing/tracking component giving the user the freedom (and obligation) to model all electronic and software components himself. This allows the user to investigate the details of the real component under development/test without the need to disclose any know how.

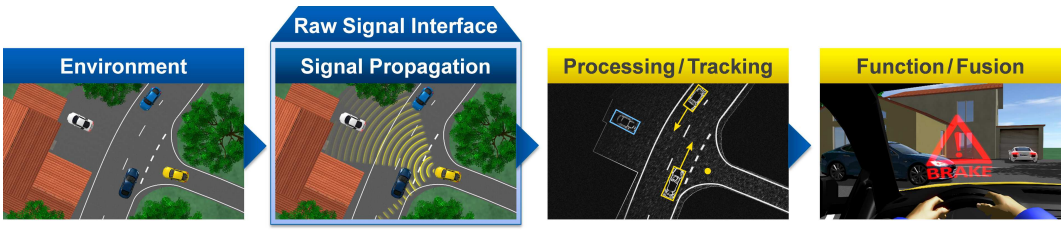


Figure 20.3: RSI for component development and testing (blue: CM, yellow: user)

20.1.4 Overview

The following table depicts the assignment of the currently available sensor models to the introduced classes. Further models of HiFi Sensors and RSIs will be unveiled with the upcoming releases.

Ideal Sensors	High Fidelity Sensors	Raw Signal Interfaces
Object Sensor	Radar Sensor	Ultrasonic RSI
Free Space Sensor	Global Navigation Sensor	Radar RSI
Traffic Sign Sensor		Camera RSI
Line Sensor		
Road Sensor		
Collision Sensor		
Slip Angle Sensor		
Inertial Sensor		

20.2 Inertial Sensor

In the vehicle and trailer at each point (chassis body, wheel carrier) an inertial sensor can be placed to measure at this point the position, velocity, rotational velocity, acceleration and rotational acceleration. Following parameters are used to define a sensor:

Sensor.Inertial.N = *Number*

Indicates the number of defined inertial sensors. Default: 0.

Sensor.Inertial.<i>.Mounting = *MountFrame*

Indicates the frame on which the inertial sensor is mounted. There are the both vehicle body frames `Fr1A, Fr1B` and all wheel carrier frames `Fr2FL...Fr2RR` available. Default: `Fr1A`.

Sensor.Inertial.<i>.pos = *x y z*

Inertial sensor position expressed in `FrD` coordinates. If the sensor is mounted on the wheel carrier, the parameters describe the position before the static equilibrium configuration.

Sensor.Inertial.<i>.rot = *x y z*

Rotation of sensor frame according to the frame on which the sensor is mounted. Rotating angles rx, ry, rz with the rotation order ZYX. Unit: [deg]. Default: 0 0 0.

Example `Sensor.Inertial.<no>.rot = 0 0 10`

Sensor.Inertial.<i>.CalcClass = *Calculation Class*

Indicates which quantities should be calculated. Following options are available:
`Global`: all velocities and accelerations are calculated in global frame `Fr0`.

`Global+Local`: all velocites and accelerations are calculated in global frame `Fr0` and in local frame on which the body sensor is mounted.

`Global+LocalNoG`: all velocites and accelerations are calculated in global frame `Fr0` and in local frame on which the body sensor is mounted. The accelerations in local frame don't consider the gravitation.

Default: `Global+Local`.

User Accessible Quantities

For additional User Accessible Quantities of the inertial sensor please refer to section 24.14.1 'Inertial Sensor' on page 795.

20.3 Slip Angle Sensor

In the vehicle and trailer at each point of the chassis body a slip angle sensor (SAngleSensor) can be placed to measure at this point the sideslip angle. Following parameters are used to define a sensor:

Sensor.SAngle.N = *Number*

Indicates the number of used slip angle sensors. Default: 1.

Sensor.SAngle.<i>.name = *Name*

Indicates the name of the slip angle sensor. Default: SS00.

Sensor.SAngle.<i>.Mounting = *MountFrame*

Indicates the frame on which the slip angle sensor is mounted. There are the both vehicle body frames `Fr1A` and `Fr1B` available. Default: `Fr1A`.

Sensor.SAngle.<i>.pos = *x y z*

Slip angle sensor position, expressed in `FrD` coordinates.

User Accessible Quantities

For additional User Accessible Quantities of the slip angle sensor please refer to section 24.14.2 'Slip Angle Sensor' on page 795.

20.4 Object Sensor

20.4.1 Introduction

The Object Sensor Module (ObjectSensor) supports interfaces to integrate various sensors used for adaptive cruise control, parking assistance and the like within CarMaker.

First an example of a modern adaptive cruise controller should show the workflow from the sensor antenna to the control algorithms.

Structure of Adaptive Cruise Controller

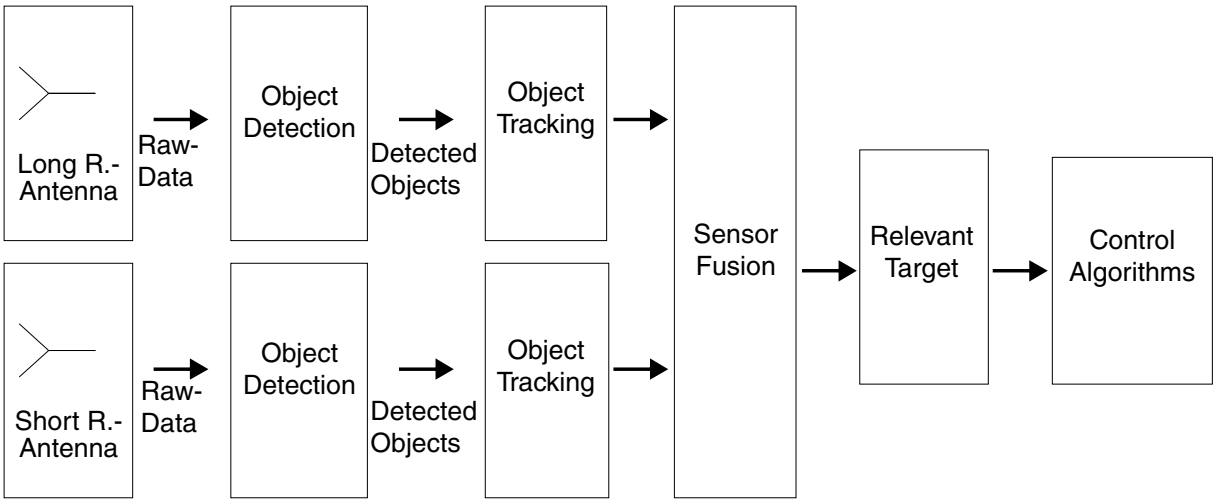


Figure 20.4: Structure of a ACC System

Figure 20.4 shows the structure of a typical adaptive cruise controller with a long range and a short range beam. Each sensor antenna delivers reflection patterns called raw data. These patterns are grouped into a list of detected objects. The object tracking algorithms filter fixed obstacles, store the remaining obstacles in a database for verification and checking. Depending on the application a lane tracking procedure is done to find targets on the vehicles driving lane, etc.

For systems with multiple sensors the data of each sensor has to be merged into a complete sensor view. This is done with the sensor fusion algorithms. Finally a relevant target can be detected and delivered to the control algorithms, e.g. the target to follow and control distance.

The steps to determine a relevant control target from the antenna raw data are regarded the most demanding tasks when developing new driver assistance systems.

20.4.2 Integration Levels

To test such a driver assistance systems with CarMaker one can think of different levels of integration.

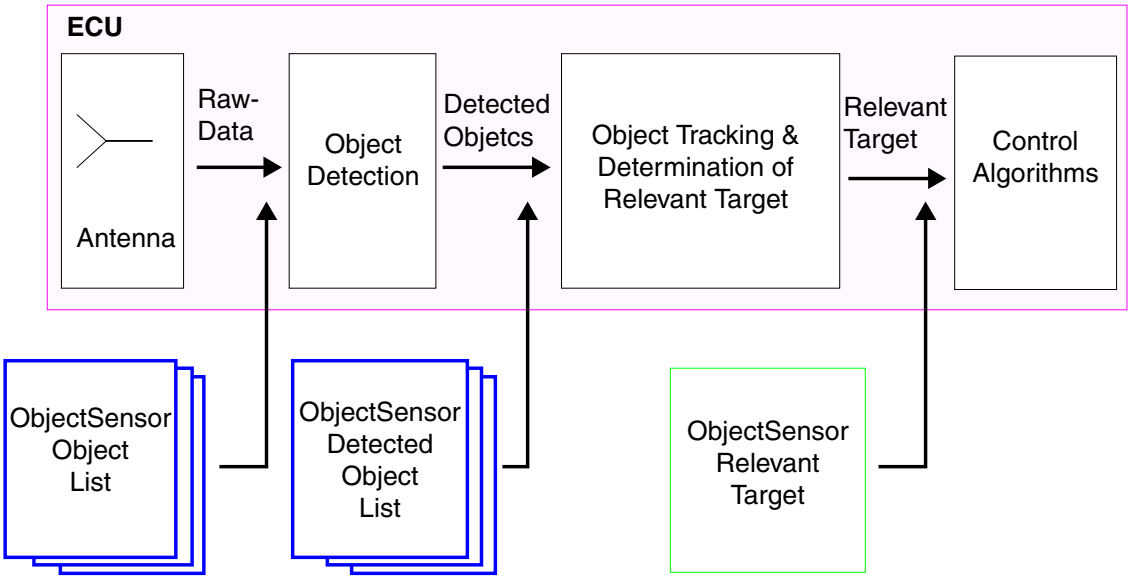


Figure 20.5: Integration Levels for ObjectSensors

Depending on where the interface within the workflow of the control unit should be, different scenarios are imaginable. Some possible applications which can be used with CarMaker are described below:

- Radar Simulation + Object Detection using provided Object List
- Object Tracking + Control Algorithm Design using provided Detected Object List
- Control Algorithm Design using provided Relevant Target

Physical Simulation (e.g. Radar) + Object Detection

If your are interrested in testing the Object Detection Algorithms based on real antenna data your hardware / ECU / prototyping system under test needs to have an input interface to pass raw data from a antenna.

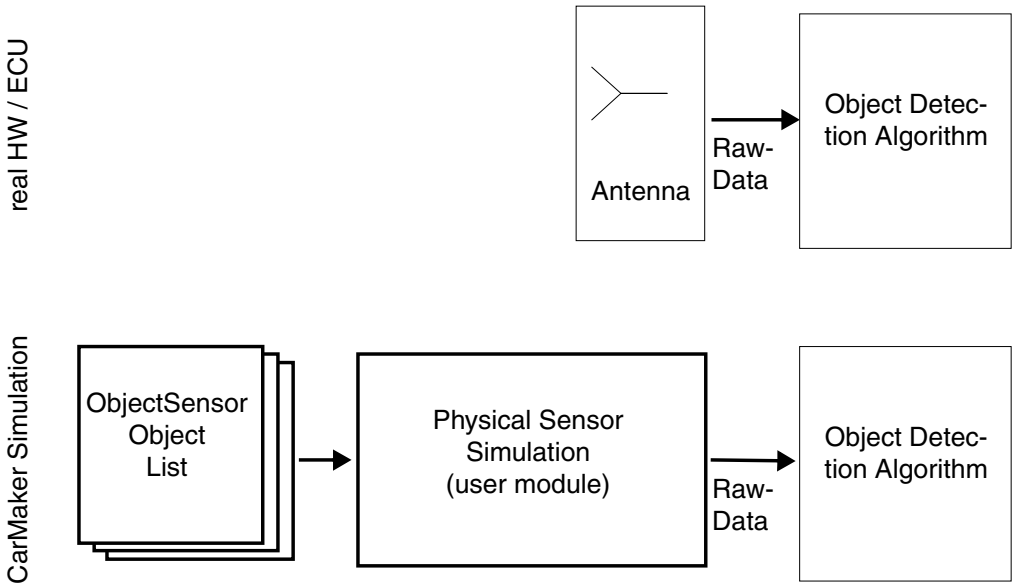


Figure 20.6: Physical Simulation and Object Detection

CarMaker cannot deliver the physical raw data from an antenna/sensor. Therefore a physical sensor simulation (e.g. radar) is needed for the specific system. Input to this simulation is the obstacle list from CarMaker which provides the sensor simulation with information listed in section 20.4.3 'Object List':

The radar simulation generates reflection patterns for each traffic obstacle simulated with CarMaker. The output should be the raw data signals as produced by the real antenna.

Object Tracking + Control Algorithm Design

If the real object detection algorithms are not of interests for your application CarMaker can provide informations about all detected objects by a certain sensor. Because there are no sensor specific attributes regarded, this approach is a simplification of the physical simulation method. Your hardware / ECU / prototyping system needs to have an input interface to pass the list of detected objects.

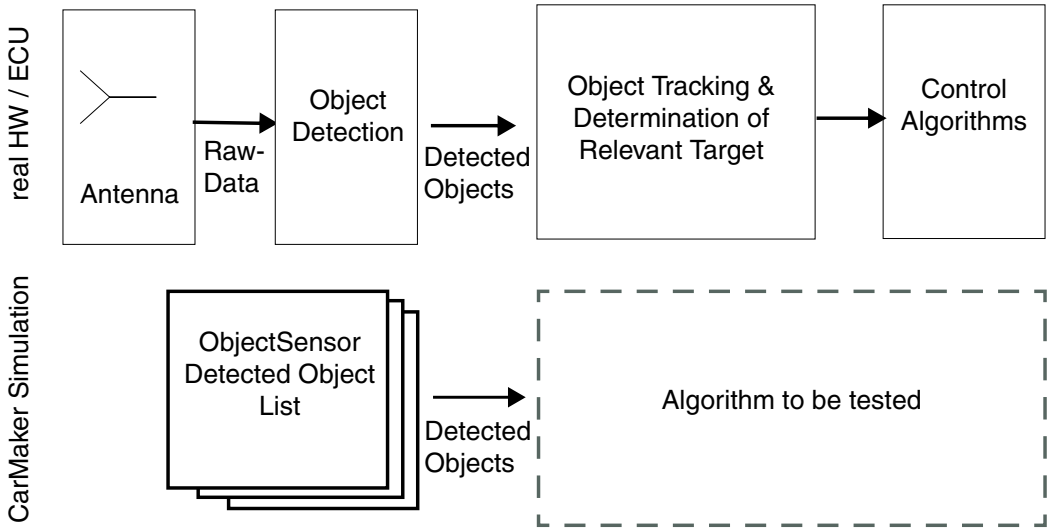


Figure 20.7: Object Tracking and Control Algorithm Design

The Object list from the CarMaker sensor interface can deliver the same quantities which would normally be generated by the obstacle detection algorithms. The full list of provided information can be found in section 20.4.3 'Object List'.

Control Algorithm Design

Another possibility is to simulate the behavior of a complete sensor up to the selection of a relevant target. This is useful for developers of control algorithms who do not want to take care of sensor aspects when they test the algorithms under development.

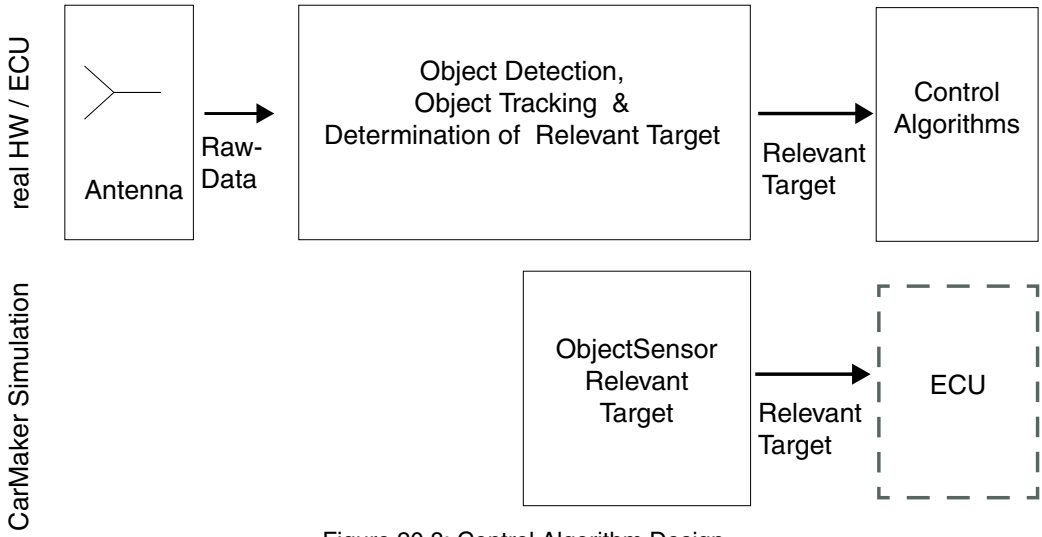


Figure 20.8: Control Algorithm Design

The relevant target is described by the same quantities as the any object of the object list when simulating in the detected objects mode. But here you get only one object, the relevant control target.

20.4.3 Object List

The CarMaker object sensor module provides for each configured sensor a object list with quantities for each traffic object relative to the view of the sensor

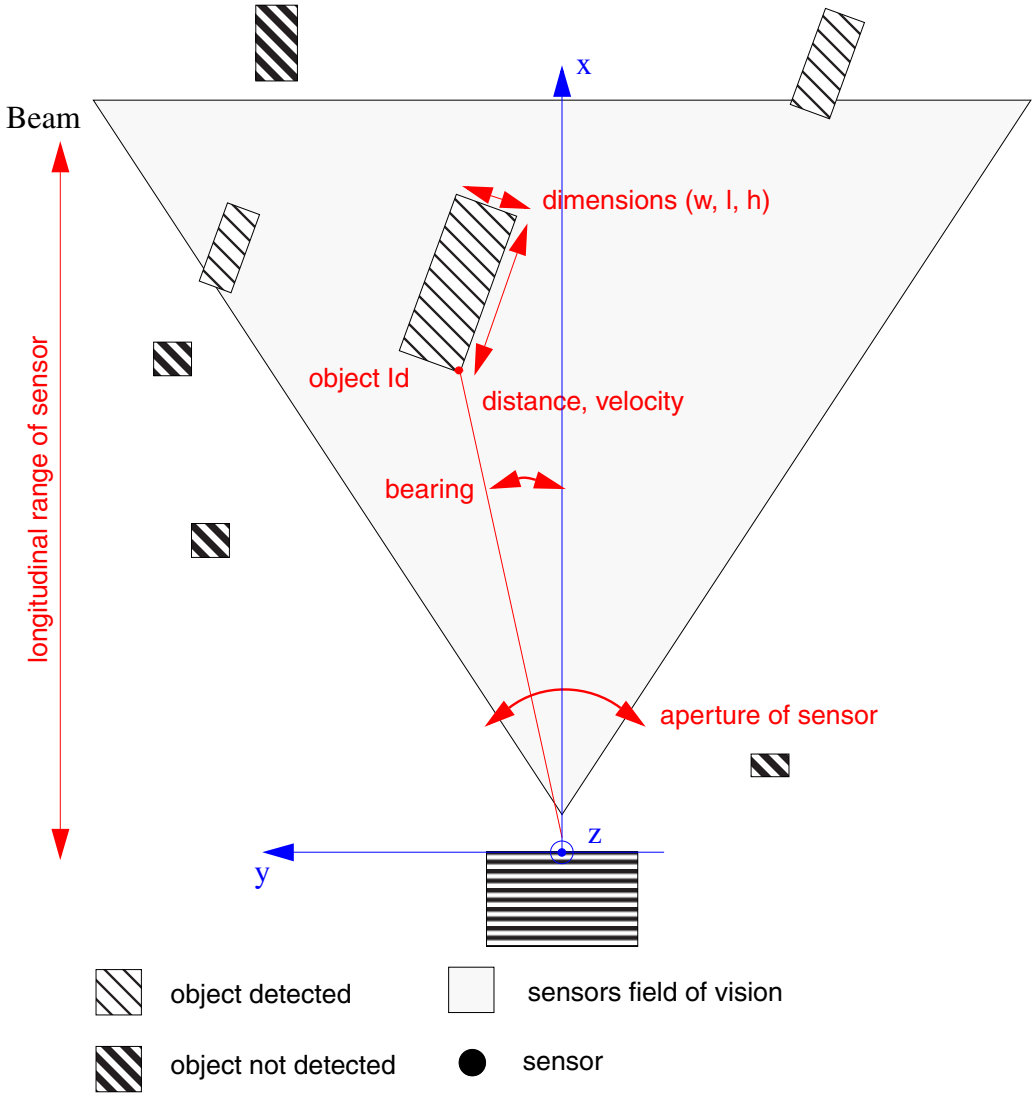


Figure 20.9: Object List

Available quantities in object list:

- object Id
- bearing (reference point and nearest point)
- relative distance and velocity (reference point and nearest point)
- relative orientation z-y-x (reference point)
- distance in x, y, z of sensor frame (reference point and nearest point)
- velocity in x, y, z of sensor frame (reference point and nearest point)
- flag: object is detected (in sensor viewing area)
- flag: object is in observation area
- incidence angles between the sensor beam and the object
- width, height, length of object and height above ground (only in C-code)
- object name (only in C-code)

CarMaker C-Code Interface to Object List

See header file *Sensor_Object.h*. The Object List can be accessed via the provided `ObjectSensor_GetObject` function.

ObjectSensor_GetObject - Get object from object list

Syntax

```
tObjectSensorObj *ObjectSensor_GetObject(int SensorNumber, int ObjectNumber)
```

Description

Returns pointer to `tObjectSensorObj` structure. If object or sensor doesn't exist, a NULL pointer is returned.

`tObjectSensorObj` structure:

```
typedef struct {
    struct tObservPoint {
        /* cartesian koordinates (sensor frame) */
        double        ds[3];        /* distance */
        double        dv[3];        /* velocity */
        double        r_zyx[3];     /* relative orientation (z-y-x) */
        /* polar coordinates */
        double        ds_p;         /* distance */
        double        dv_p;         /* velocity */
        double        alpha_p;      /* bearing azimuth */
        double        theta_p;      /* bearing elevation */
    } RefPnt, NearPnt;

    /* object specific */
    char          dtct;         /* object in defined sensor range */
    char          InLane;       /* object is within driving lane */
    char          obsv;         /* object on observation area */
    double        incangle[2];  /* incidence angles */
    double        w, h, l;      /* dimensions */
    double        zOff;         /* height of object ref. point above ground */
    int           ObjId;        /* global object identification nr. */
    char          Name[TRF_MAMESIZE]; /* traffic object name */

   /* projection on image area */
   double         ImgArea_NearP; /* projection of the nearest point */
   double         ImgArea_LeftP; /* projection of the left point */
   double         ImgArea_RightP; /* projection of the right point */
} tObjectSensorObj;
```

Example

```
tObjectSensorObj *tgt;
for (i=0; i<Traffic.nObjs; i++) {
   tgt = ObjectSensor_GetObject(0, i);
   if (tgt != NULL)
      Log("Distance to traffic object: %5.1f\n", tgt->RefPnt.ds_p);
}
```

Data Dictionary

See section 20.4.8 'User Accessible Quantities' on page 633.

20.4.4 Calculation class

Searching of the nearest Object Point

The object sensor module calculates the relative quantities in the reference object point and in the nearest object point if the calculation class *CalcClass = NearestPoint* is selected.

There are several algorithms implemented to detect a correct point:

- If the object is located outside of the sensors field of vision, the sensor takes the nearest point (a corner or the perpendicular point).
- If the object is located inside of the sensors field of vision, even partially, the nearest point corresponds to the nearest located point of the detected object. This could be a point along the object surface or a corner.

. Figure 20.10 demonstrates how the sensor module detects the nearest object point by rotating the sensor range and keeping the object fix:

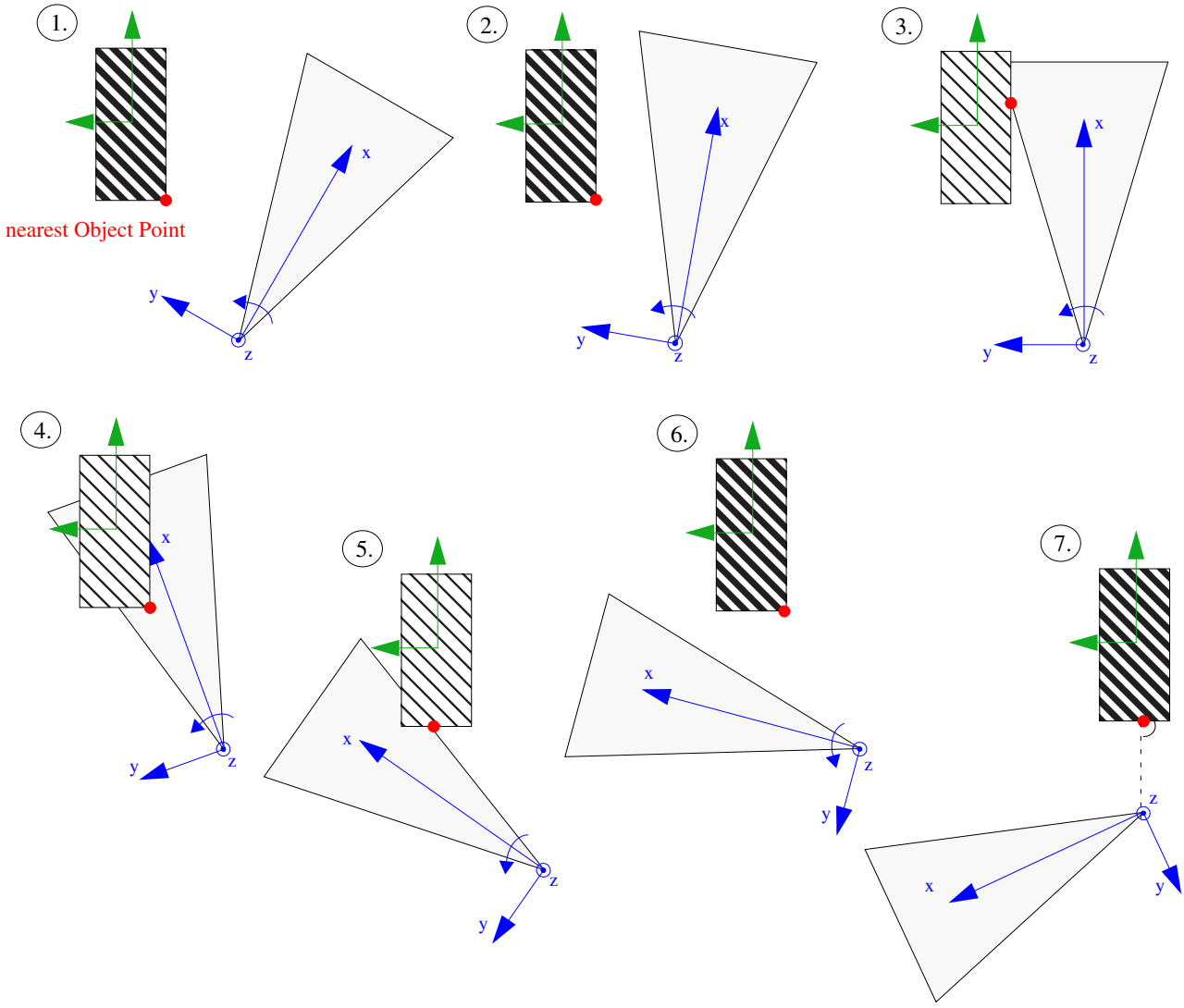


Figure 20.10: Object Point Detection

Image Area

If the object is detected the sensor module calculates the angle of incidence alpha and beta on the object point:

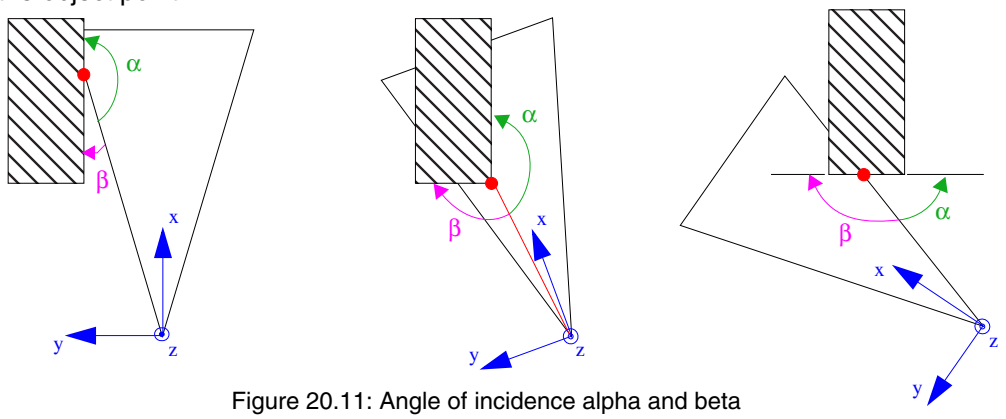


Figure 20.11: Angle of incidence alpha and beta

In addition to the calculation of angle of incidence, the object sensor module calculates the degree of coverage for each object. This functionality can be used for calculating if an object is hidding another one and/or to calculate reflections.

For this the nearest, the left and the right detected point of the object are projected on the horizontal plane (xy-plane of the sensor) of the sensor image area, see Figure 20.12. The position on the image area can be in a range between -1 and 1. At the position zero the object point is in the middle of the sensors field of vision.

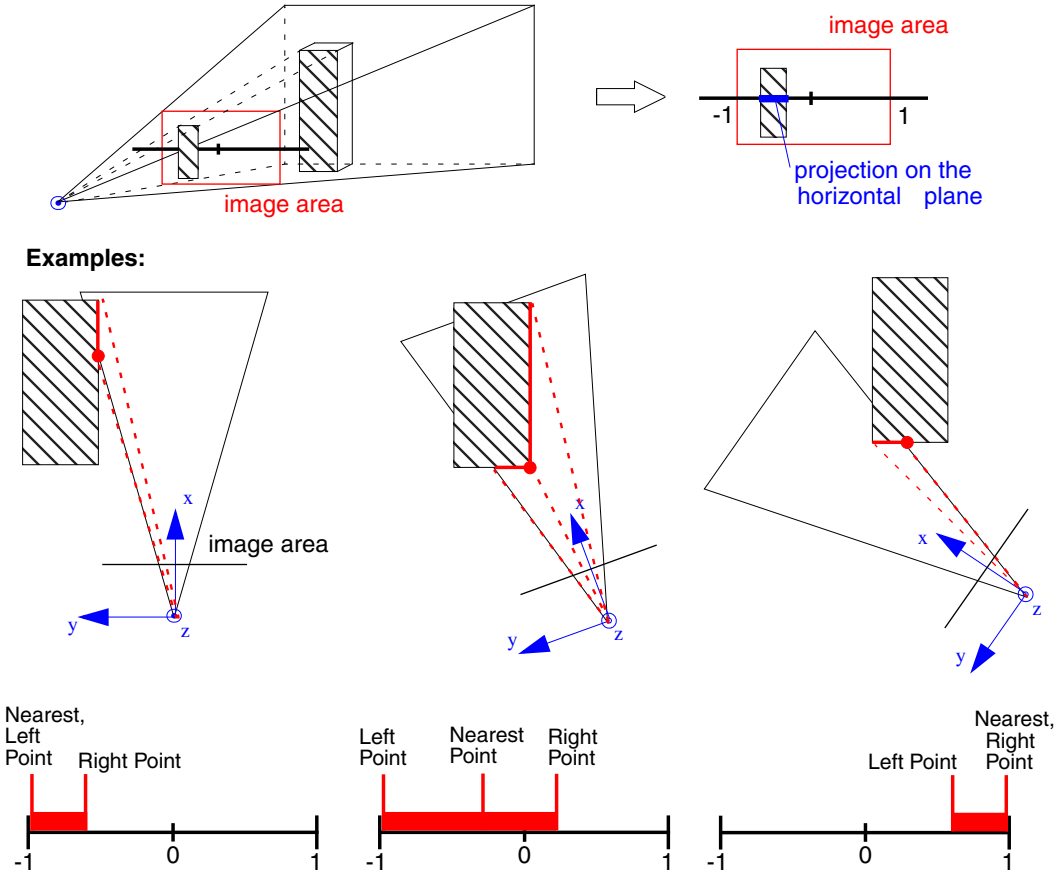


Figure 20.12: Projection on the sensor image area

If the object is located outside of the sensors field of vision (not detected), the two angles of incidence are set to zero and the projected position of the object points on the image area are set to -2.

20.4.5 Target Selection

This section describes how the sensor calculates a relevant target.

This is an additional feature of the module and only available if the sensor is configured to do this kind of calculations. The object list as described in section 20.4.3 is always available.

The target selection algorithms are fairly simple and not meant to have all features of a real sensor. The sensor can be configured (see section 20.4.7 'Parametrization of Object Sensor') in two modes:

- Nearest Object:
  With this mode the closest of all visible objects of the sensor is taken as relevant target. This mode is useful for parking distance controllers and alike.
- Nearest Object in Path:
  This mode finds the closest object within a range of the estimated trajectory of the vehicle. This mode is especially useful for adaptive cruise control applications where only objects are considered as relevant targets if they shroud the vehicles driving lane.

Nearest Object

Each sensor can detect a closest object.

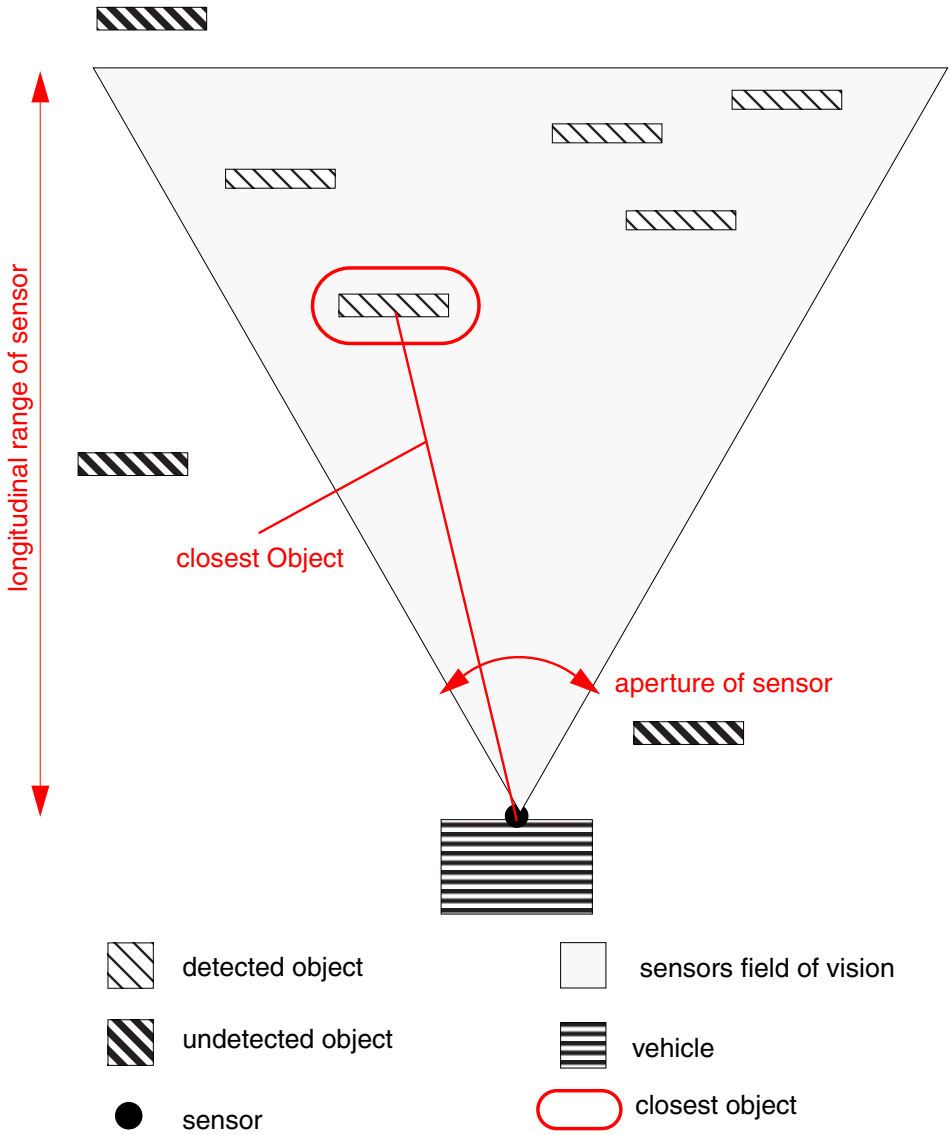


Figure 20.13: Finding closest object

The sensor module iterates with predefined update rate over all objects to find the closest object and calculates the distance from this object to the sensor. Depending on the calculation class the distance is referring to the reference point (*CalcClass = ReferencePoint*) or the nearest point (*CalcClass = NearestPoint or ImageArea*). The object with the smallest overall distance is the relevant target.

If two or more objects have the same distance, the first detected object will be taken into account.

Nearest Object in Path (Vehicle Trajectory Identification)

The detection of a relevant target for a ACC Sensor is done by investigation of the current vehicles trajectory the width of its driving lane and the distance to the object.

After looping over all objects on the vehicles driving lane the relevant target is calculated by using the closest object algorithm as described in section 'Nearest Object'.

The driving lane detection algorithm is described below:.

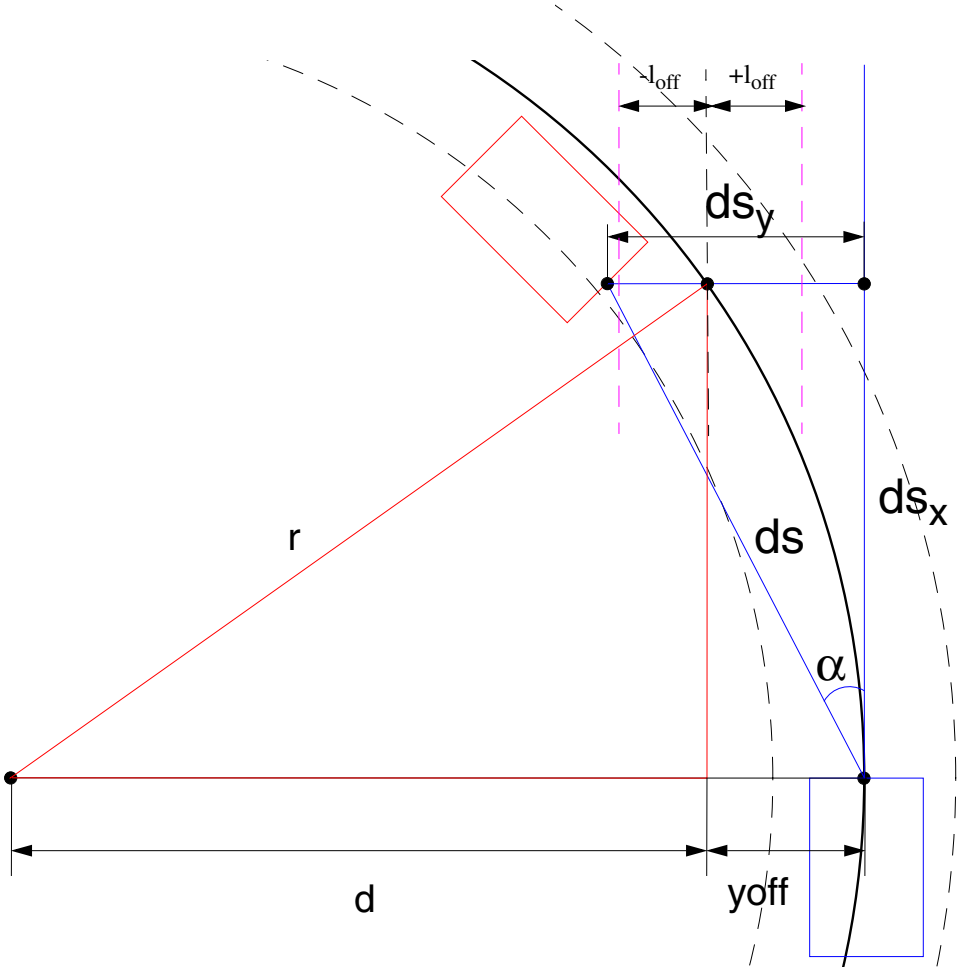


Figure 20.14: Vehicle Trajectory Identification

ds	projected distance to target vehicle
α	angle of target vehicle in sensor frame
ds_x	component in X direction of projected distance in sensor frame
ds_y	component in Y direction of projected distance in sensor frame
r	turning radius
y_{off}	imaginary vehicle offset in sensor frame at target position
l_{off}	vehicles lane width / 2
a_y	lateral acceleration of vehicle
v	vehicle speed

Turning radius:

$$r = \frac{v^2}{|a_y|} \quad \text{(EQ 447)}$$

Components in sensor frame:

$$ds_x = ds \cdot \cos(\alpha) \quad \text{(EQ 448)}$$

$$ds_y = ds \cdot \sin(\alpha) \quad \text{(EQ 449)}$$

Imaginary vehicle offset in sensor frame at target position:

$$y_{off} = r - d = (r - \sqrt{r^2 - ds_x{}^2}) \cdot sign(a_y) \quad \text{(EQ 450)}$$

$$l_{off} = \text{vehicles lane width}/2 + \text{lane offset} \quad \text{(EQ 451)}$$

Check if target vehicle is within vehicles trajectories limits:

$$(y_{off} - l_{off} < ds_y) \cap (ds_y < y_{off} + l_{off}) \quad \text{(EQ 452)}$$

This target selection mode is intended for the path detection in front of the ego-vehicle. This means that the rotation of sensor frame according to vehicle frame should be zero.

CarMaker Interfaces to Object Detection

C-Code Interface

See header file *Sensor_Object.h*. The relevant target can be accessed e.g via the `ObjectSensor[i].relvTarget` variable. This points to a `tObjectSensorObj` structure.

Example

```
double dist;
dist = ObjectSensor[i].relvTarget.ds_p;
```

Data Dictionary

See section 20.4.8 'User Accessible Quantities' on page 633.

20.4.6 Overview of Object Sensor Calculation

For the calculation of the sensor quantities, CarMaker calls the function *ObjectSensor_Calc()*, defined in the header file *Sensor_Object.h*. Figure 20.15 gives an overview of this function:

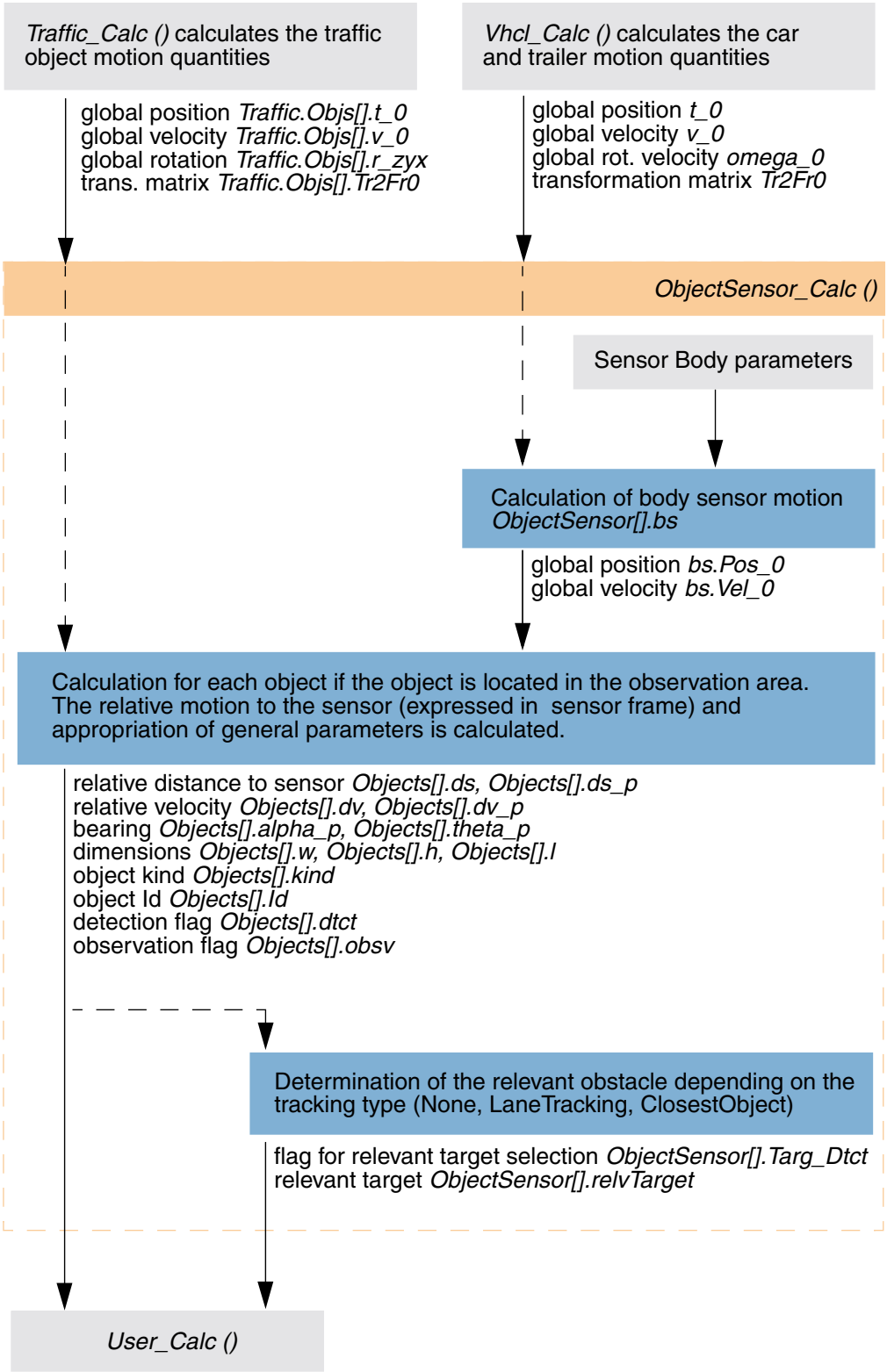


Figure 20.15: Overview of the *ObjectSensor_Calc* () function

20.4.7 Parametrization of Object Sensor

The object sensors are parametrized in the vehicle file and in the trailer file.

Sensor.Object.N

Specifies the total number of vehicle and trailer object sensors to be used. Default: 0.

Sensor.Object.ObservRadius

Specifies the radius of the observation area [m]. Default: 500m.
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.

Sensor.Object.<no>.name

A name for each sensor can be given. Default: DA00.

Example `Sensor.Object.0.name = RadarL`

Sensor.Object.<no>.tracking

Type of target selection. Can be `LaneTracking` (nearest object in path), `ClosestObject` or `None`.

Example `Sensor.Object.0.tracking = LaneTracking`

Sensor.Object.<no>.CalcClass

Calculation class: `ReferencePoint`, `RelativeAngles`, `NearestPoint` or `ImageArea`.
Default: `NearestPoint`.

`ReferencePoint`	Calculation of relative quantities in reference point
`RelativeAngles`	`ReferencePoint` + Calculation of relative angles in reference point
`NearestPoint`	`RelativeAngles` + Calculation of relative quantities in nearest point
`ImageArea`	`NearestPoint` + Calculation of incidence angles and projections on image area

Example `Sensor.Object.0.CalcClass = ImageArea`

Sensor.Object.<no>.Mounting

Indicates the frame on which the sensor is mounted. There are the vehicle body frames `Fr1A` and `Fr1B` available. Default: `Fr1A`.

Example `Sensor.Object.0.Mounting = Fr1B`

Sensor.Object.<no>.alpha

Horizontal aperture of sensor beam [deg]. Maximal possible value is 180 deg.

Example `Sensor.Object.0.alpha = 16.0`

Sensor.Object.<no>.theta

Vertical aperture of sensor beam [deg]. Maximal possible value is 180 deg.

Example `Sensor.Object.0.theta = 4.0`

Sensor.Object.<no>.pos

Position of the sensor in vehicle frame (FrD). Coordinates x, y, z [m].

Example `Sensor.Object.0.pos = 4.0 0.0 0.5`

Sensor.Object.<no>.range

Longitudinal range of the sensor [m].

Example `Sensor.Object.0.range = 150`

Sensor.Object.<no>.rot

Rotation of sensor frame according to vehicle frame. This specifies the viewing direction of the sensor. Rotating angles rx, ry, rz [deg]. The rotation order is ZYX.

Example `Sensor.Object.0.rot = 0 0 90`

Sensor.Object.<no>.UpdRate

Specifies with which rate the sensor detects the environment and updates the signals [Hz]. Default: 60.

Example `Sensor.Object.0.UpdRate = 30`

Sensor.Object.<no>.nCycleOffset

Specifies the number of cycles after this the sensor begins with the calculation of the signals. Default: 0.

Example `Sensor.Object.0.nCycleOffset = 5`

Sensor.Object.<no>.ShowDDict

If set to 1 additional quantities for all traffic objects are shown in DataDictionary. Default: 0.

Sensor.Object.<no>.movietheme

Used for visualization of the sensor with *IPGMovie*. Possible values are:

- `NotVisible`
- `Cyan`
- `Blue`
- `Pink`
- `Red`
- `Yellow`
- `Orange`
- `Green`
- `Magenta`

Example `Sensor.Object.0.movietheme = Blue`

Sensor.Object.<no>.ExtMotion

If set to 1 external relative sensor travel and rotation are considered. Default: 0.
The external sensor travel and rotation can be set directly in the global C-variable `ObjectSensor[number].t_ext/rot_zyx_ext` or via Direct Variable Access on the quantities `Sensor.Object.<name>.[tx,ty,tz,rx,ry,rz]_ext`.

Sensor.Object.<no>.LaneWidth

Optional. Specifies the lane width if the target selection type 'LaneTracking' is selected. Unit: [m]. Default value is vehicle width + 0.4m safety margin at left and right side.

Object Sensor of IPGDriver

For observation of the traffic around the vehicle, the *IPGDriver* uses the sensor module. The two last sensor elements after the user specified elements are reserved and predefined internally for the driver module. The Id can be obtained with the function *ObjectSensor_FindIndexForName*() using the sensor names *Driver_FRWD* and *Driver_OTR*.

The default parameters of the driver object sensors are:

Parameter	Value
Driver sensor for observation of ahead and oncoming traffic	
Sensor.Object.*Id_fwrd*.name	Driver_FRWD
Sensor.Object.*Id_fwrd*.range	700
Sensor.Object.*Id_fwrd*.alpha	60
Sensor.Object.*Id_fwrd*.theta	45
Sensor.Object.*Id_fwrd*.pos	Position of PoI (Point of Interest)
Sensor.Object.*Id_fwrd*.rot	0 0 0
Sensor.Object.*Id_fwrd*.UpdRate	500
Sensor.Object.*Id_fwrd*.nCycleOffset	0
Sensor.Object.*Id_fwrd*.tracking	LaneTracking
Sensor.Object.*Id_fwrd*.CalcClass	NearestPoint
Sensor.Object.*Id_fwrd*.movietheme	NotVisible
Driver sensor for overtaking traffic on right side	
Sensor.Object.*Id_otr*.name	Driver_OTR
Sensor.Object.*Id_otr*.range	25
Sensor.Object.*Id_otr*.alpha	120
Sensor.Object.*Id_otr*.theta	5
Sensor.Object.*Id_otr*.pos	Position of PoI (Point of Interest)
Sensor.Object.*Id_otr*.rot	0 0 -60
Sensor.Object.*Id_otr*.UpdRate	500
Sensor.Object.*Id_otr*.nCycleOffset	0
Sensor.Object.*Id_otr*.tracking	None
Sensor.Object.*Id_otr*.CalcClass	NearestPoint
Sensor.Object.*Id_otr*.movietheme	NotVisible

Each internal driver sensor can be manipulated by defining an object sensor with the name *Driver_FRWD* or *Driver_OTR*.

20.4.8 User Accessible Quantities

Relevant Target Quantities

Each sensor provides information about its relevant target if there is one detected.

Only if Sensor.Object.<nm>.relvTgt.dtct=1 (<nm> stands for the sensor name) the remaining quantities carry valid informations for this particular sensor. If Sensor.Object.<nm>.relvTgt.dtct=0 the remaining quantities are signed over with zero and should not be considered.
Please refer to section 'Relevant Target' on page 796.

Object List Quantities

The object list quantities are only visible in the data dictionary if the parameter Sensor.Object.<nm>.ShowDDict is set to 1.
Please refer to section 'Object List' on page 797.

20.4.9 Visualization in IPGMovie

Each object sensor is automatically visualized in *IPGMovie*. Its position, aperture, range and direction is shown according to the parameters given for this sensor.

Figure 20.16: Visualization of different sensors in *IPGMovie*

Visualization of Detected Objects in IPGMovie

If the object is displayed in *IPGMovie* by a box, in the case of detection the box changes the color to blue (detected object) or yellow (detected and relevant object, respectively). If the object is displayed in *IPGMovie* by a geometry defined in an object file *XXX.obj*, in the case of detection the object geometry will be enclosed by a transparent box.

20.5 Free Space Sensor

20.5.1 Introduction

The Free Space Sensor Module (FSpaceSensor) is an extended object sensor 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 stereo camera and capture around the vehicle the free and the occupied space, which could be further used for an avoidance maneuver assistant.

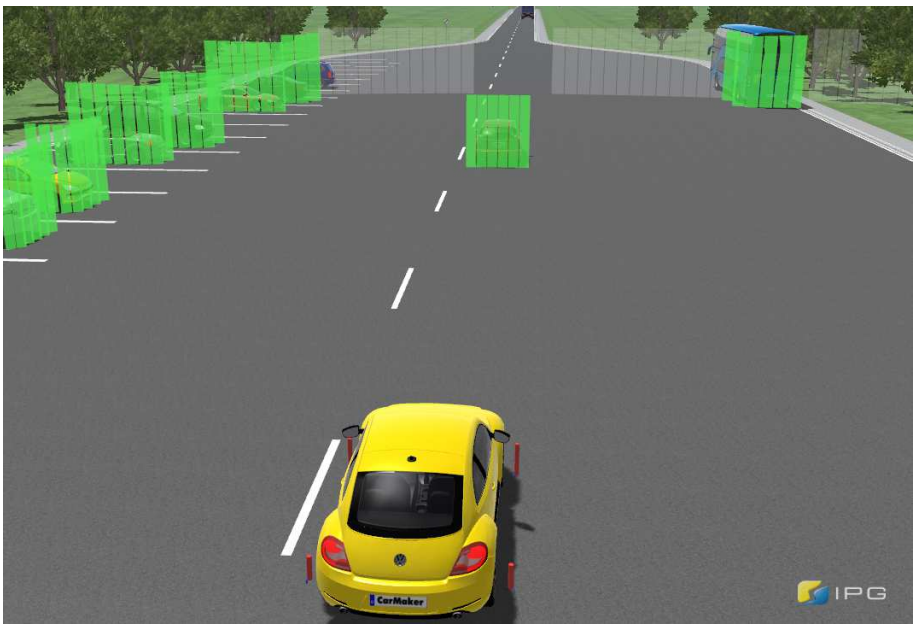


Figure 20.17: Free Space Sensor

20.5.2 Parametrization of Free Space Sensor

The free space sensors are parameterized in the vehicle file and in the trailer file.

Sensor.FSpace.N

Specifies the total number of vehicle and trailer free space sensors to be used. Default: 0.

Sensor.FSpace.<no>.name

A name for each sensor can be given. Default: FS00.

Example `Sensor.FSpace.0.name = FS_Front`

Sensor.FSpace.<no>.Mounting

Indicates the frame on which the sensor is mounted. There are the both vehicle body frames `Fr1A` and `Fr1B` available. Default: `Fr1A`.

Example `Sensor.FSpace.0.Mounting = Fr1B`

Sensor.FSpace.<no>.alpha

Horizontal aperture of sensor beam [deg]. Maximal possible value is 180 deg.

Example `Sensor.FSpace.0.alpha = 16.0`

Sensor.FSpace.<no>.theta

Vertical aperture of sensor beam [deg]. Maximal possible value is 180 deg.

Example `Sensor.FSpace.0.theta = 4.0`

Sensor.FSpace.<no>.pos

Position of the sensor in vehicle frame (FrD). Coordinates x, y, z [m].

Example `Sensor.FSpace.0.pos = 4.0 0.0 0.5`

Sensor.FSpace.<no>.range

Longitudinal range of the sensor [m].

Example `Sensor.FSpace.0.range = 150`

Sensor.FSpace.<no>.rot

Rotation of sensor frame according to vehicle frame. This specifies the viewing direction of the sensor. Rotating angles rx, ry, rz [deg]. The rotation order is ZYX.

Example `Sensor.FSpace.0.rot = 0 0 90`

Sensor.FSpace.<no>.nHorSegm

Specifies the number of horizontal sensor segments. Default: 110.
The horizontal aperture of each sensor segment is equal.

Example `Sensor.FSpace.0.nHorSegm = 250`

Sensor.FSpace.<no>.nVerSegm

Specifies the number of vertical sensor segments. Default: 4.
The vertical aperture of each sensor segment is equal.

Sensor.FSpace.<no>.UpdRate

Specifies with which rate the sensor detects the environment and updates the signals [Hz]. Default: 25.

Example `Sensor.FSpace.0.UpdRate = 30`

Sensor.FSpace.<no>.nCycleOffset

Specifies the number of cycles after this the sensor begins with the calculation of the signals. Default: 0.

Example `Sensor.FSpace.0.nCycleOffset = 5`

Sensor.FSpace.<no>.ExtMotion

If set to 1, external relative sensor travel and rotation are considered. Default: 0.
The external sensor travel and rotation can be set directly in the global C-variable `FSpaceSensor[number].t_ext/rot_zyx_ext` or via Direct Variable Access on the quantities `Sensor.FSpace.<name>.[tx,ty,tz,rx,ry,rz]_ext`.

Sensor.FSpace.<no>.movietheme

Used for the sensor visualization with IPGMovie. Possible values are:

- `NotVisible`
- `Cyan`
- `Blue`
- `Pink`
- `Red`
- `Yellow`
- `Orange`
- `Green`
- `Magenta`

20.5.3 User Accessible Quantities

Please refer to section 24.14.4 'Free Space Sensor' on page 797.

20.6 Free Space Sensor Plus

20.6.1 Introduction

The Free Space Sensor Plus can be imagined like an extended Free Space Sensor as described in section 20.5 'Free Space Sensor' on page 634 with a few differences. Internally it uses a completely separate algorithm since it does its calculations on the GPU. This allows us to not only detect traffic objects but also every other 3d-object (e.g. road decoration objects or the road itself) as they are displayed in IPGMovie. This means that the evaluation of the object for detection of it's nearest point is not limited to it's bounding-box, but rather the exact 3d-geometry consisting of several triangles / polygons is analyzed. Since this requires lots of calculations, we use massive parallelization to identify the point of interest (nearest point or point with strongest reflection) for each sensor beam.

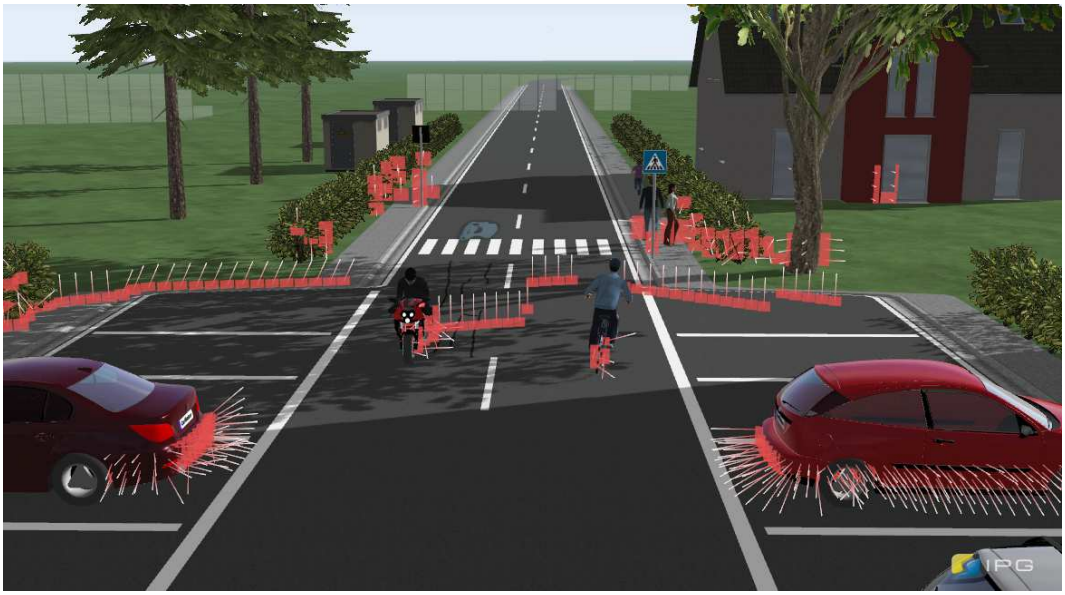


Figure 20.18: Free Space Sensor Plus

20.6.2 Functional description

Like the classical Free Space Sensor, the Free Space Sensor Plus also subdivides the sensor's field of view into equiangular horizontal and vertical segments like you can see in the picture below. Each segment consists of a parameterizable number of sub-segments. Since internally we render various (distorted) images from the sensor's point of view (with the parameterized field of view), technically those sub-segments are pixels. Out of these pixels within one segment, the one pixel that fits a filter criteria (nearest distance or strongest reflection) is chosen and its distance value is assigned to the segment's corresponding sensor beam before a quantity is generated for this. We provide the following quantities of the 3D-object surface point that was hit by the sensor beam:

- Distance
- Normal Vector
- Radial Velocity
- Object Id
- Material Id

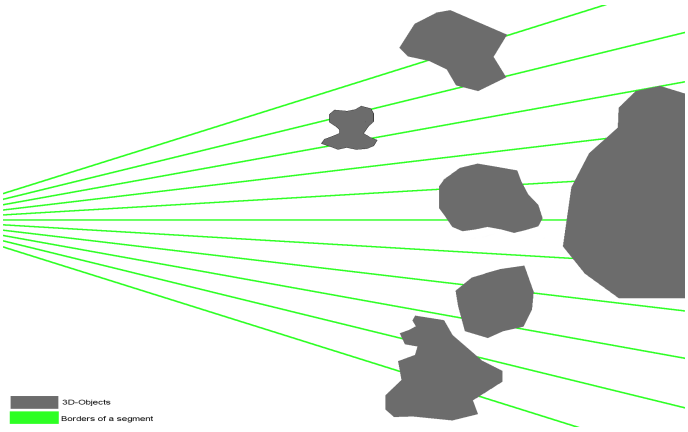


Figure 20.19: Subdivision of the field of view

20.6.3 Point filters

You have the possibility to choose between the following two point filters as a selection method for the relevant pixel (which represents the point on the 3D-object's surface) within one segment:

- Nearest
- Strongest

The following chapters explain these point filters.

Nearest

This filter method identifys the point on the 3D-object's surface (represented with a pixel in various rendered images) within one segment, that has the shortest distance to the sensor's position. If there are more than one pixels within one segment, that have the exact same distance, the one is chosen that is more centered within the segment.

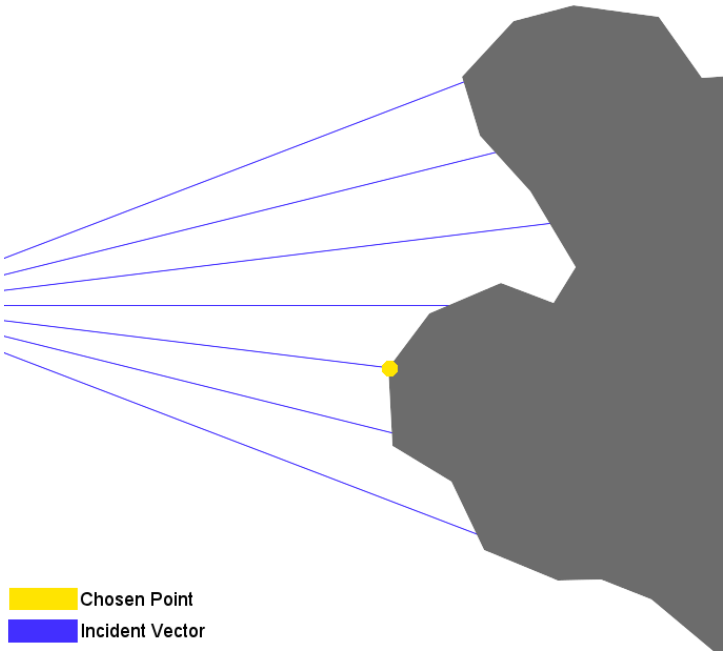


Figure 20.20: Point Filter Nearest

Strongest

This filter method identifys the point on the 3d-objects surface with the smallest reflection angle. To calculate this reflection angle, we compute the appropriate incident vector for each pixel within each segment. This incident vector is reflected along the surface normal of the 3d-object. The point within each segment is chosen for which this reflection angle is minimal. If there are multiple pixels with the exact same minimal reflection angle, the pixel (point on 3d-object surface) with the shortest distance to the sensor's position is chosen. If there are still multiple pixels that have the exact same minimal reflection angle and the exact same distance to the sensor's position, the pixel is chosen that is more centered within the segment.

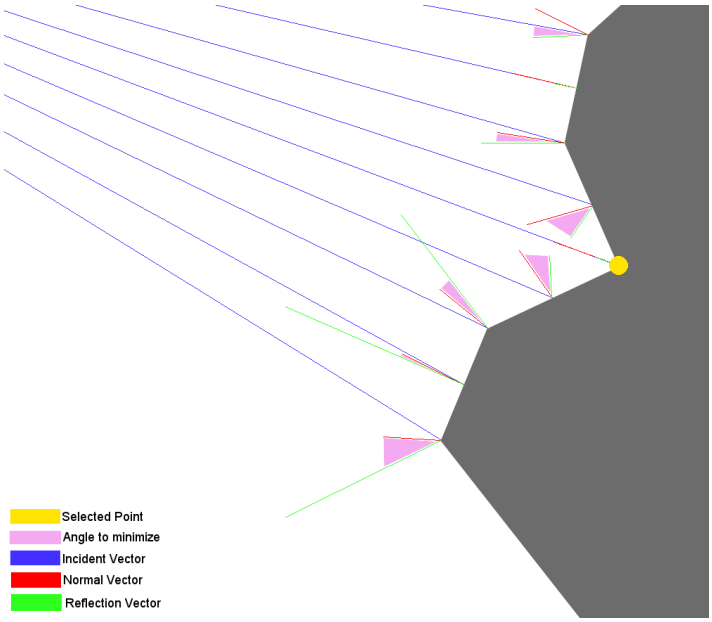


Figure 20.21: Point Filter

20.6.4 Parametrization of Free Space Sensor Plus

The free space plus sensors are parameterized in the vehicle file and in the trailer file. In the following only differences to the classical Free Space Sensor and additional Free Space Sensor Plus specific parameters are mentioned.

Sensor.FSpace.<no>.alpha=	**float**

Horizontal field of view of sensor [deg]. Maximal possible value is 270 deg. Default: 110 deg.

Sensor.FSpace.<no>.theta=	**float**

Vertical field of view of sensor [deg]. Maximal possible value is 90 deg. Default: 3.4 deg.

Sensor.FSpace.Plus.Latency = float float

Relative min. and max. latency multiplied with cycle time results in a range for quantity update (cf. Figure 20.22). A physical sensor has a latency close to cycle time. This results in a parameter value of 1.0 1.0. Default: 1.0 1.0.

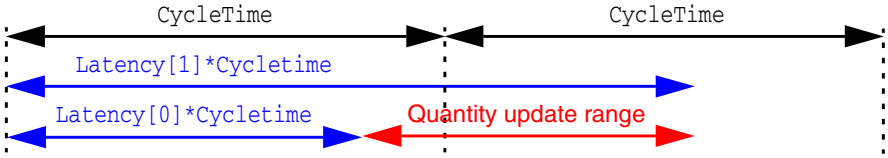


Figure 20.22: Quantity update range.

This parameter has huge impact on sensor performance!
The latency value is used internally to handle synchronization of CPU and GPU calculation. If the two values differ, the synchronization is weak, the simulation faster, but jitter will occur. Different synchronization modes can be adjusted:

- Quantity update without jitter [1.0 1.0]
- Quantity update with weak synchronization, allowing for faster simulation [1.0 2.0]
- Quantity update as soon as possible [0.0 1.0]

The enforcement of synchronization is handled differently depending on the simulation type. If the calculation does not finish in time, the simulation will be slowed down in CarMaker/Office whereas a realtime violation occurs in CarMaker/HIL. New sensor calculation is started after quantity update.

Sensor.FSpace.Plus.Quantities = bool

Enables creating quantities. For a hugh number of segments disabling is highly recommended, especially in realtime simulations. Default: 0.

Sensor.FSpace.<no>.Plus.range_min= float

Minimum range of the sensor [m]. Everything which is closer than that, is not detected

Sensor.FSpace.<no>.Plus.Active= bool

Bool Value that defines if Free Space Sensor Plus or classical Free Space algorithm is used

Sensor.FSpace.<no>.Plus.Filter= enum

Defines which point filter is used for selection of the relevant point within a segment. Possible values "nearest" or "strongest"

Sensor.FSpace.<no>.Plus.nSegmResHor= int

Defines the horizontal number of sub-segments / points within one segment. In order to avoid subsampling effects, the parameter has to be adjusted depending on Sensor.FSpace.<no>.nHorSegm. Default: 5.

Sensor.FSpace.<no>.Plus.nSegmResVer= int

Defines the vertical number of sub-segments / points within one segment. In order to avoid subsampling effects, the parameter has to be adjusted depending on Sensor.FSpace.<no>.nVerSegm. In the default configuration with nSegmVer = 4 this leads to an rather high value. Default: 50.

Sensor.FSpace.<no>.Plus.AddQuants= enum

Defines which quantities are calculated / created per segment. One or more of the possible values (separated by a space character) are allowed:

- "NormalVec": Calculates the normal vector of the detected point.
- "RadialVec": Calculates the relative radial velocity of the nearest point (distance change rate).
- "ObjectId": Returns the global object ID of the detected object.
- "MaterialId": Provides the material ID in the detection point according to the material definitions in the parameterization of the Environment module.
- "CosIncid": Calculates the cosine of the angle between the sensor ray and the object's surface normal at hit point.

20.7 Traffic Sign Sensor

20.7.1 Introduction

The Traffic Sign Sensor Module (TSignSensor) detects in a specified sensor view (defined by horizontal/vertical aperture angles and range) all or some selected traffic signs like an idealized camera.

The sensor collects all signs in a sensor view range, verifies if they are facing to the sensor and sorts the signs with ascending distance. Additionally the sensor provides sign information like identification, supplement sign information, sign description and two sign values (e.g. velocity).

Figure 20.23: Traffic Sign Sensor

20.7.2 Parametrization of Traffic Sign Sensor

The traffic sign sensors are parameterized in the vehicle file.

Sensor.TSign.N

Specifies the total number of vehicle traffic sign sensors to be used. Default: 0.

Sensor.TSign.<no>.name

A name for each sensor can be given. Default: TS00.

Example `Sensor.TSign.0.name = TS_Front`

Sensor.TSign.<no>.Mounting

Indicates the frame on which the sensor is mounted. There are the both vehicle body frames `Fr1A` and `Fr1B` available. Default: `Fr1A`.

Example `Sensor.TSign.0.Mounting = Fr1B`

Sensor.TSign.<no>.alpha

Horizontal aperture of sensor view [deg]. Maximum possible value is 90 deg.

Example `Sensor.TSign.0.alpha = 16.0`

Sensor.TSign.<no>.theta

Vertical aperture of sensor view [deg]. Maximum possible value is 90 deg.

Example `Sensor.TSign.0.theta = 4.0`

Sensor.TSign.<no>.pos

Position of the sensor in vehicle design frame FrD. Coordinates x, y, z [m].

Example `Sensor.TSign.0.pos = 4.0 0.0 0.5`

Sensor.TSign.<no>.range

Longitudinal range of the sensor [m].

Example `Sensor.TSign.0.range = 150`

Sensor.TSign.<no>.rot

Rotation of sensor frame according to mounted frame. This specifies the viewing direction of the sensor. Rotating angles rx, ry, rz [deg]. The rotation order is ZYX.

Example `Sensor.TSign.0.rot = 0 0 90`

Sensor.TSign.<no>.UpdRate

Specifies with which rate the sensor detects the traffic signs and updates the signals [Hz]. Default: 30.

Example `Sensor.TSign.0.UpdRate = 30`

Sensor.TSign.<no>.nCycleOffset

Specifies the number of cycles after this the sensor begins with the calculation of the signals. Default: 0.

Example
```
Sensor.TSign.0.nCycleOffset = 5
```

Sensor.TSign.<no>.ExtMotion

If set to 1, external relative sensor travel and rotation are considered. Default: 0.
The external sensor travel and rotation can be set directly in the global C-variable `TSignSensor[number].t_ext/rot_zyx_ext` or via Direct Variable Access on the quantities `Sensor.TSign.<name>.[tx,ty,tz,rx,ry,rz]_ext`.

Sensor.TSign.<no>.PosInFrB

If set to1, the relative sign position is expressed in the body frame FrB, on which the sensor is mounted. If set to 0, the relative sign position is expressed in sensor frame FrS.

Sensor.TSign.<no>.RefPoint

Position of the reference point in vehicle design frame FrD, in which the relative sign position is calculated if *PosInFrB* is set to 1. Coordinates x, y, z [m].

Sensor.TSign.<no>.SignSelection:

Specifies which traffic sign kinds the sensor should detect.
If set *All*, the sensor searchs for all sign kinds. The sensor can capture at once at maximum 40 signs.

Example
```
Sensor.TSign.0.SignSelection:
   All
```

The sensor can also detect only some specified signs like *Stop* or *Velocity*. The maximum number for selected sign kinds is limited to 10. The sensor can capture at once for each selected sign kind 4 traffic signs.

Example
```
Sensor.TSign.0.SignSelection:
   Stop
   Caution
   CurveL
   GradeUp
```

20.7.3 User Accessible Quantities

Please refer to section 24.14.5 'Traffic Sign Sensor' on page 798.

20.8 Line Sensor

20.8.1 Introduction

The Line Sensor Module detects in a specified sensor view (defined by horizontal/vertical aperture angles and range) road markings and traffic barriers like a idealized camera.

The sensor collects all lines and barriers in a sensor view and sorts into left and right lines with ascending lateral distance to a specified frame point. The specified frame point could be the sensor origin point in the sensor frame or any reference point in the vehicle frame Fr1A or Fr1B. Additionally the sensor provides line information like identification, color code, type, width and height (for traffic barrier). Furthermore in the global headerfile `Sensor_Line.h` the absolute and relative position of the line points are available for visualization and lane/path prediction.

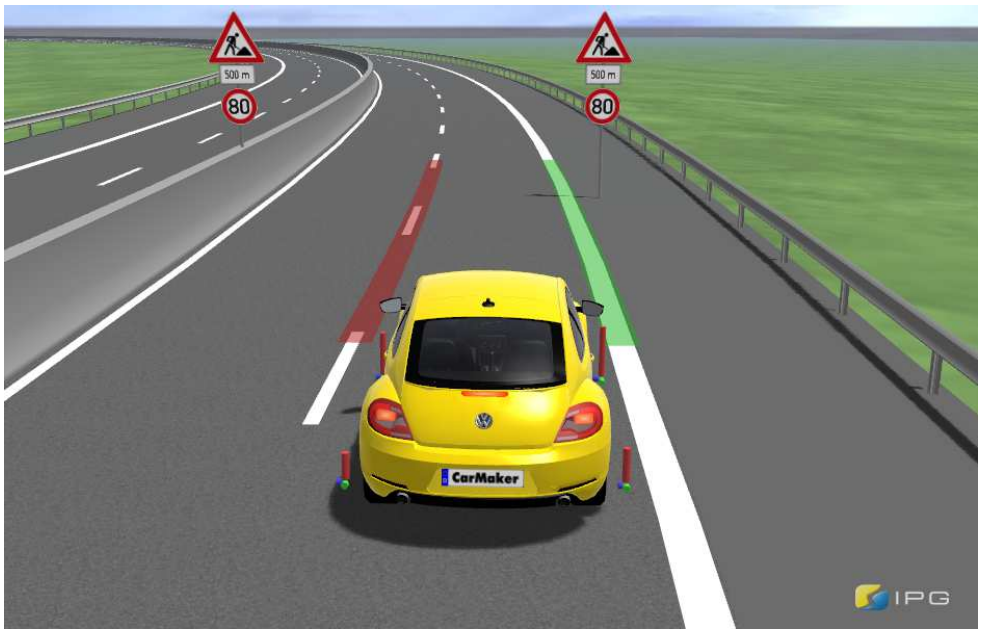


Figure 20.24: Line Sensor

20.8.2 Detection of the Lines

For the detection of the lines and barriers, a shape with seven points on the road (orange shape in the following figures) is identified supposing the road as flat area around the vehicle. The sensor view is divided in five vertical and three horizontal areas. A grid is defined by crossing the vertical and horizontal areas (white points), which is used to calculate the shape on the road (see second picture in Figure 20.25) by linear intersection with the road area.

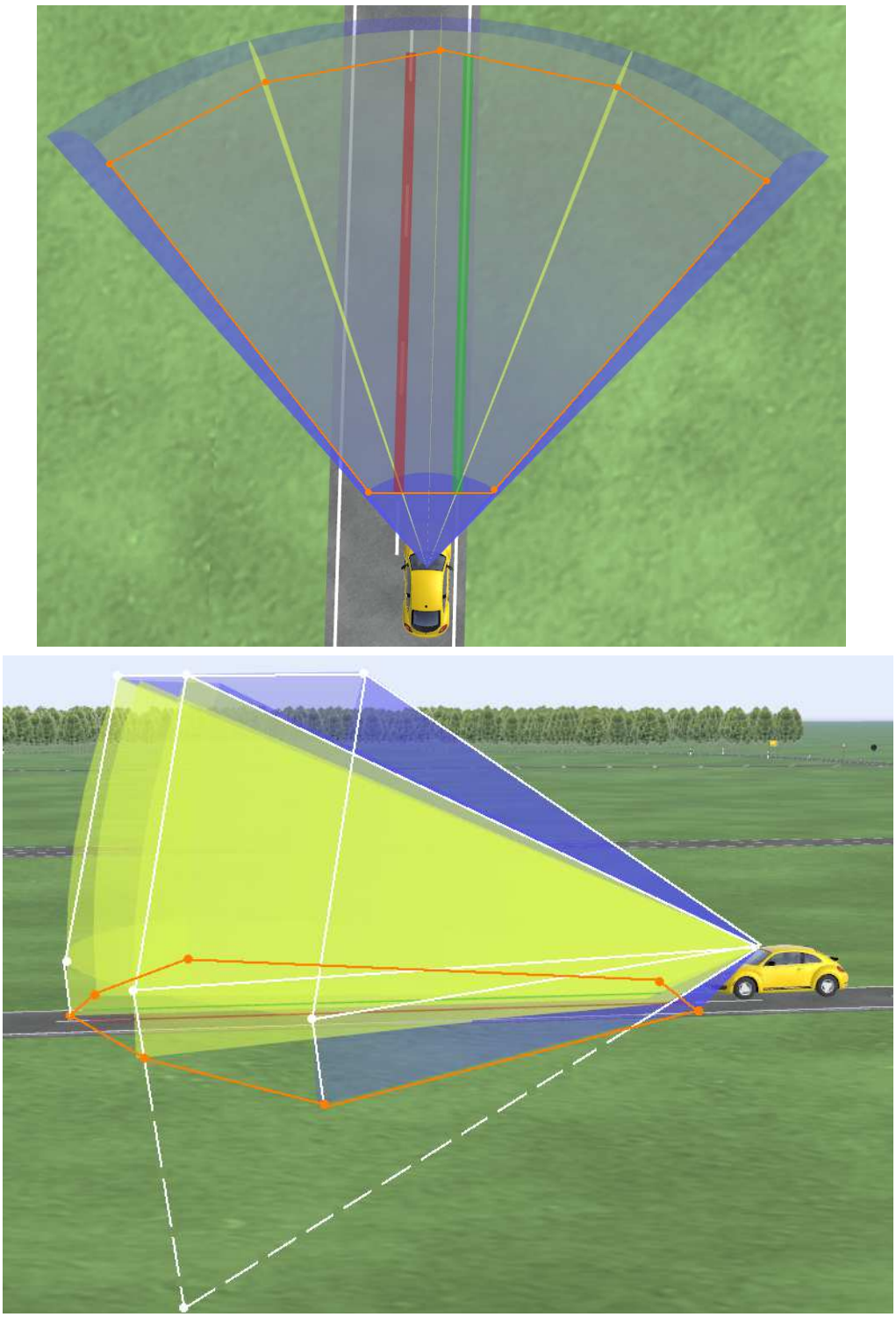


Figure 20.25: Line Sensor Modell

In front of the sensor the shape is modelled only with two points due to performance gain in the road library. Thus leads to a bigger area shape in front of the vehicle (white area in Figure 20.26) as expected and consequently to more detected line points. This area error depends on the vertical sensor position, on vertical and horizontal aperture angles.

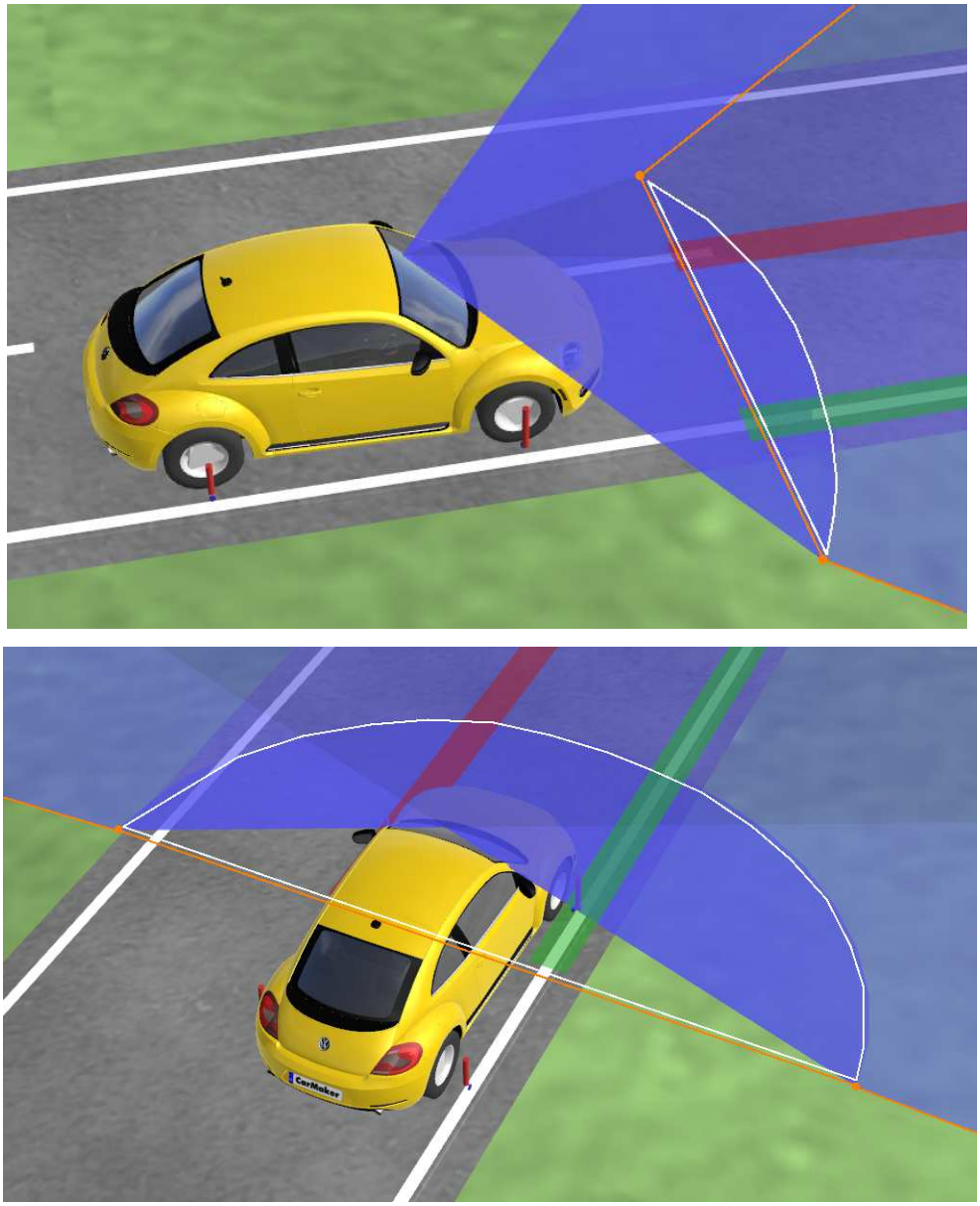


Figure 20.26: Line Sensor with 80 and 180deg horizontal view

20.8.3 Parametrization of Line Sensor

The line sensors are parameterized in the vehicle file.

Sensor.Line.N

Specifies the total number of vehicle Line Sensors to be used. Default: 0.

Sensor.Line.<no>.name

A name for each sensor can be given. Default: LN00.

Example `Sensor.Line.0.name = LN_Front`

Sensor.Line.<no>.Mounting

Indicates the frame on which the sensor is mounted. There are the both vehicle body frames `Fr1A` and `Fr1B` available. Default: `Fr1A`.

Example `Sensor.Line.0.Mounting = Fr1B`

Sensor.Line.<no>.range

Longitudinal range of the sensor [m].

Example `Sensor.Line.0.range = 150`

Sensor.Line.<no>.pos =*x* *y* *z*

Position of the sensor in vehicle design frame FrD. Coordinates x, y, z [m].

Example `Sensor.Line.0.pos = 4.0 0.0 0.5`

Sensor.Line.<no>.rot= *Rotx* *Roty* *Rotz*

Rotation of sensor frame according to mounted frame. This specifies the viewing direction of the sensor. Rotating angles rx, ry, rz [deg]. The rotation order is ZYX.

Example `Sensor.Line.0.rot = 0 0 90`

Sensor.Line.<no>.alpha = *Value*

Horizontal aperture of sensor view [deg]. Maximum possible value is 180 deg.

Example `Sensor.Line.0.alpha = 90`

Sensor.Line.<no>.theta = *Value*

Vertical aperture of sensor view [deg]. Maximum possible value is 180 deg.

Example `Sensor.Line.0.theta = 60`

Sensor.Line.<no>.UpdRate

Specifies with which rate the sensor detects the lines and updates the signals [Hz]. Default: 30.

Example `Sensor.Line.0.UpdRate = 30`

Sensor.Line.<no>.nCycleOffset

Specifies the number of cycles after this the sensor begins with the calculation of the signals. Default: 0.

Example `Sensor.Line.0.nCycleOffset = 5`

Sensor.Line.<no>.ExtMotion

If set to 1, external relative sensor travel and rotation are considered. Default: 0.
The external sensor travel and rotation can be set directly in the global C-variable `LineSensor[number].t_ext/rot_zyx_ext` or via Direct Variable Access on the quantities `Sensor.Line.<name>.[tx,ty,tz,rx,ry,rz]_ext`.

Sensor.Line.<no>.PosInFrB = *bool*

Specifies if the line points in the global Line Sensor headerfile are expressed and sorted relating to the body frame (PosInFrB=1) on which the sensor is mounted (e.g.: vehicle frame Fr1A) or to the sensor frame (PosInFrB=0; default).

Sensor.Line.<no>.RefPoint = *x* *y* *z*

Position of the reference point in vehicle frame FrD. Coordinates x, y, z [m].
This point is used for the calculation of the relative line points position if relating to the body frame (PosInFrB=1). Unit: [m]. Default: 0 0 0.

20.8.4 User Accessible Quantities

Please refer to section 24.14.6 'Line Sensor' on page 799.

20.9 Road Sensor

20.9.1 Introduction

The task of a road sensor (RoadSensor) is to determine at a predefined preview distance (horizon) important road specific attributes like the road curvature, road marker attributes (speed limit), the longitudinal and lateral slope and relative information like the deviation distance and deviation angle. This information can support in applications like lane keeping assistance, lane departure warning, autonomous driving, sign detection, energy management, prescanning, optimization of fuel consumption or detection of wheel lifting.
The following table illustrates the different applications and their possibly required RoadSensor informations:

	LK	LDW	AD	SD	EM	PS	WLD	PT
	Lane Keeping	Lane Departure Warning	Autonom. Driving	Sign Detection	Energy Manage-ment	Prescan	Wheel Lifting Detection	Powertrain
road curvature			X					
longitudinal slope, lateral slope					X			X
deviation angle, deviation distance	X	X	X					
Lane information: lane width,lane number, current passing lane	X	X	X					
global road point position x,y,z						X	X	
road marker attributes (speed limit)			X	X	X			X

20.9.2 Theory of the sensor prevision

Preview mode

Due to the various possible applications and required road information, there is a preview mode to select in which way the road sensor should search the previewed road point. The different modes are:

- *AlongRoute*: Search the previewed road point along the route and path centerline. This mode delivers the following information: curvature, long. slope, lat. slope, deviation angle and distance, global road point position for route and path; lane information.
- *AlongVhcl*: Search the previewed road point along the vehicle direction. This mode delivers the following information: road normal vector, global road point position.
- *OnlyRoadMarker*: Search the road marker along the road centerline. This mode delivers only the road marker attributes.

Previewed road point along route centerline

To find the previewed road point along the route and path centerline, at first the road sensor, located in the middle of the front axle, is projected in the global xy-plane on the route centerline. Starting at the projected point, the previewed road point is reached by following the route centerline with the defined preview distance, see Figure 20.27. For the preview point on path the determined preview point on route is projected to the path.

An algorithm is implemented to consider the route end and if the route is a circuit. If necessary the previewed road point is limited to the route end.

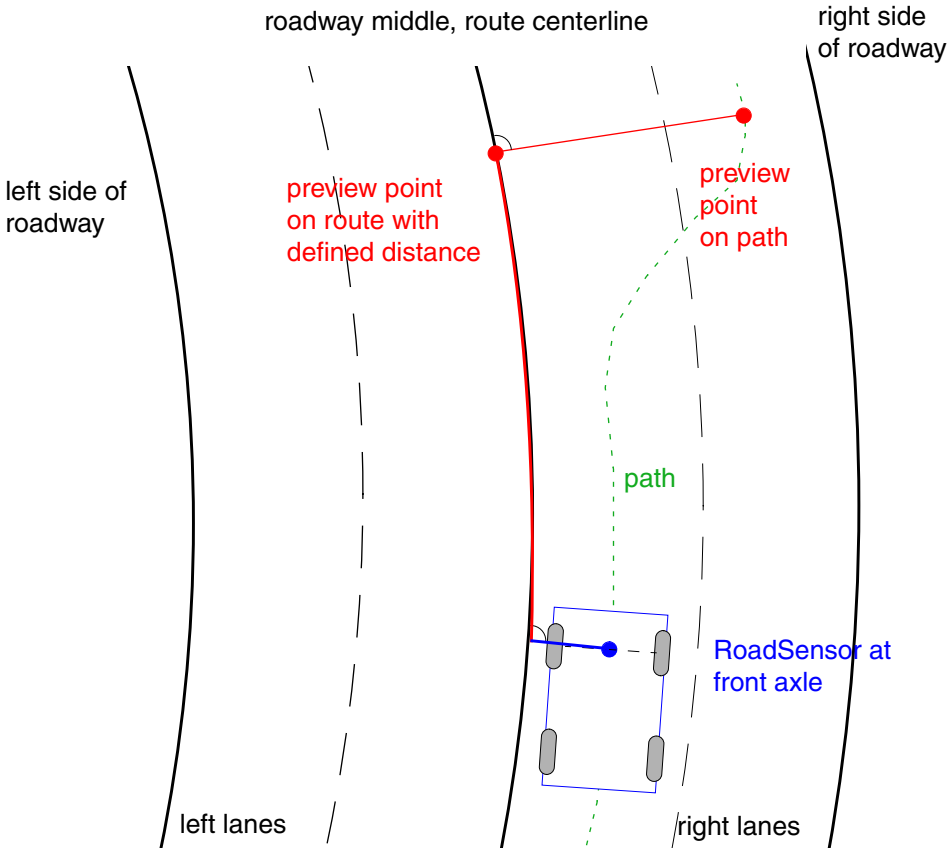


Figure 20.27: Searching the previewed point along the route and path centerline

At the previewed road point on route or path the road sensor delivers information about all lanes (lane id, width, type, lateral offset to route centerline) on left and right side of the roadway. This information is only accessible in the global struct *RoadSensor*. If driving on a valid path, the sensor modul determines at preview point additionally (c-variables and quantities) the information on current lane, and the lanes on left and right side (see Figure 20.28).

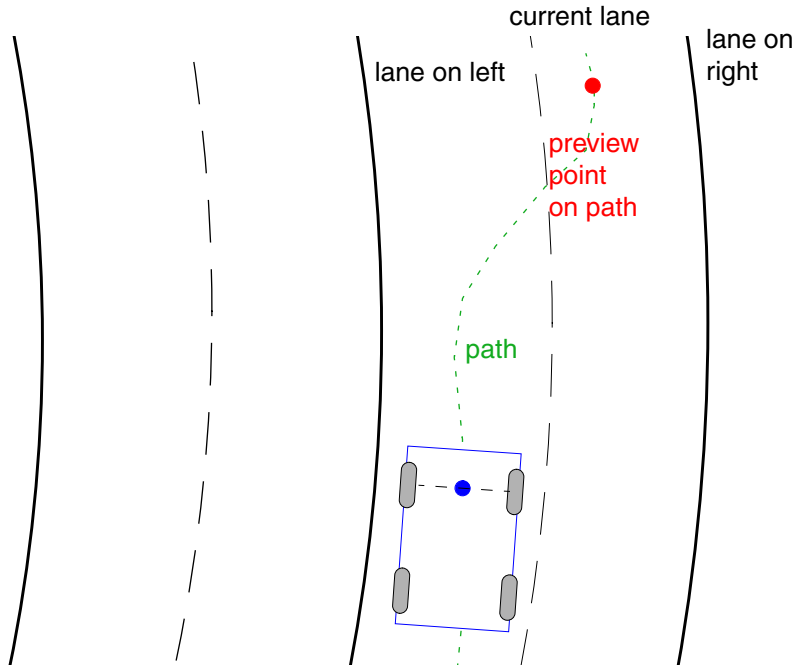


Figure 20.28: Lane information at preview point on path

The curvature in xy-plane of the route and path at the preview point is calculated by the IPGRoad library since the route and path are handled in the road library.

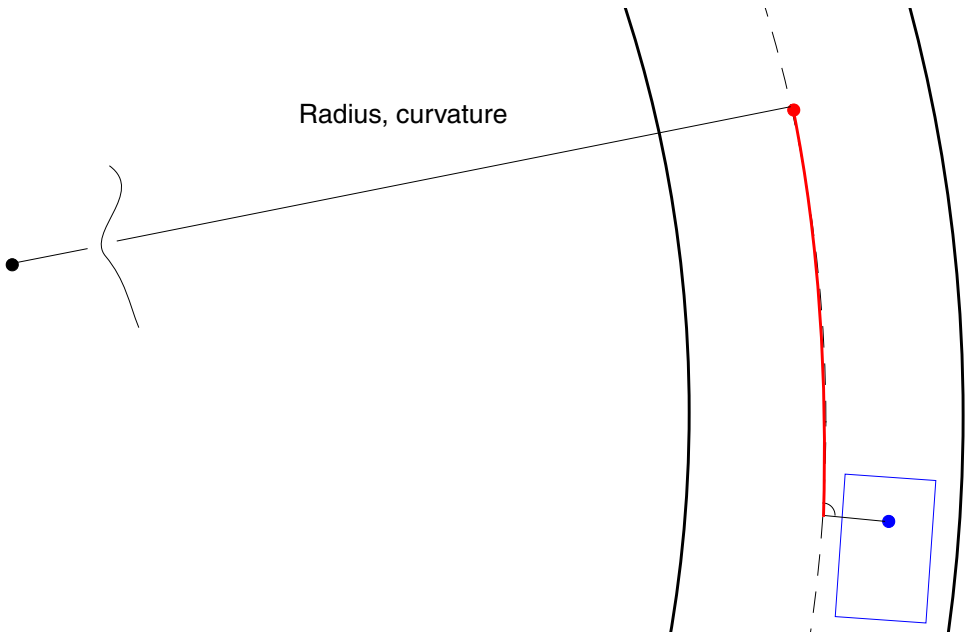


Figure 20.29: Route curvature

Calculation of the deviation distance and the deviation angle

The deviation distance is the lateral offset of the road sensor projected on the centerline of the route and path at the previewed point, see Figure 20.30 and Figure 20.31. Positive distance means left to the centerline. Deviation angle is the relative angle of the vehicle direction refered to the route and path direction at the previewed point in the global xy-plane.

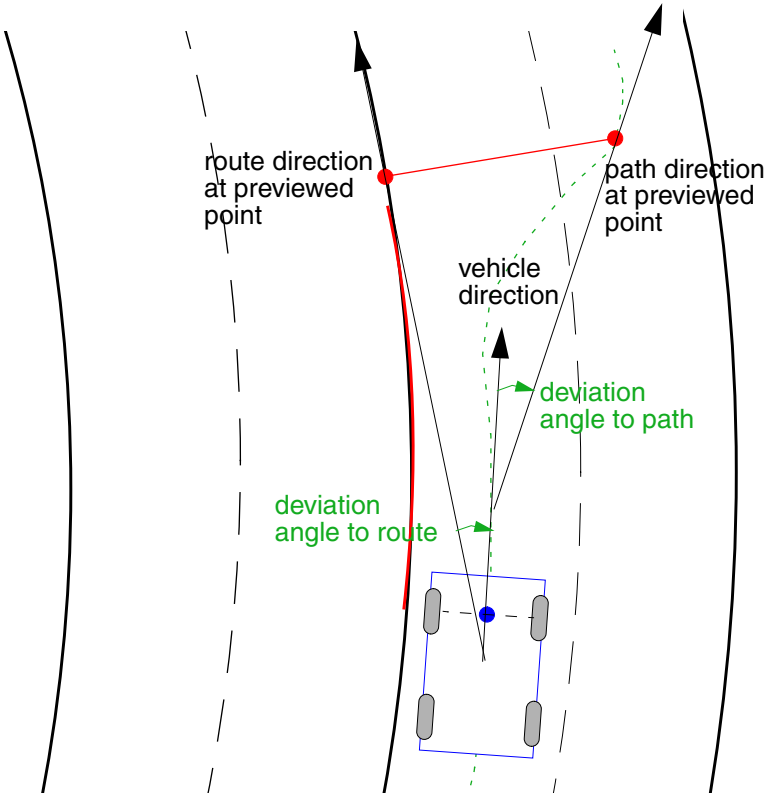


Figure 20.30: Deviation angle on route and path

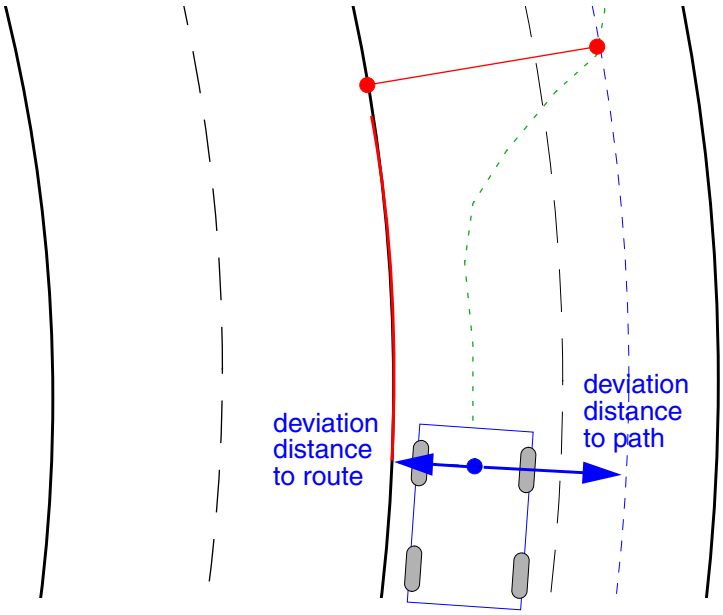


Figure 20.31: Deviation distance on route and path

Previewed road point along vehicle direction

Using this preview mode, the RoadSensor position in the vehicle can be defined anywhere relative to the vehicle frame. By default the sensor searches the road previewed point along the vehicle direction with the defined preview distance. However, it is possible to rotate the preview direction referring to the vehicle direction by an angle around the vehicle's z-axis, see Figure 20.32. In this way it is possible, if required, to 'preview' the road laterally or behind the vehicle.

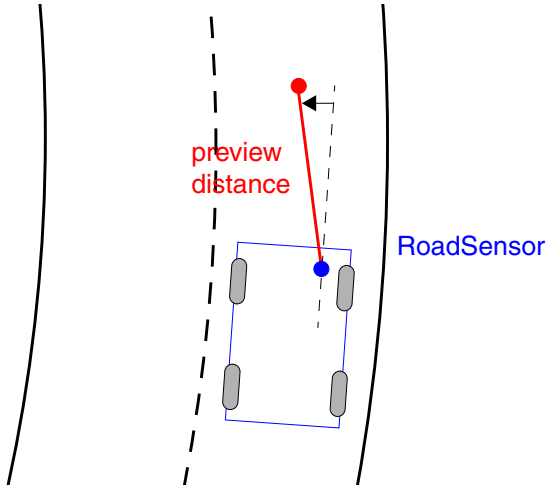


Figure 20.32: Searching the previewed road point along vehicle direction

Depending on the preview distance and the rotation angle around the z-axis, the preview point could be outside the road. For this case the flag *onRoad* is set to zero and the road information is not updated.

20.9.3 Parameterization of Road Sensor

The road sensors are parameterized in the vehicle infofile.

General Parameters

Sensor.Road.N = ***Value***

Specifies the total number of road sensors to be used. Default: 0.

Sensor.Road.<no>.name = ***NameString***

A name for each sensor can be given. Default: RP00.

Example `Sensor.Road.<no>.name = RP_1`

Sensor.Road.<no>.PreviewDist = *Value*

Specifies the preview distance. Default: 0.0. Unit: [m].

Example `Sensor.Road.<no>.PreviewDist = 100`

Sensor.Road.<no>.PreviewMode = *ModeString*

Specifies the preview mode. Possible modes are:

`AlongRoute`	Preview along the route and path centerline
`AlongVhcl`	Preview along the vehicle direction
`OnlyRoadMarker`	Preview along the road centerline only for road marker

Default: `AlongRoute`.

Example `Sensor.Road.<no>.PreviewMode = AlongVhcl`

Sensor.Road.<no>.ConsiderBumps =*Bool*

Indicates if for the preview point detection the road bumps (e.g: cylinder, beam, wave) should be considered. Default: 0.

Example `Sensor.Road.<no>.ConsiderBumps = 1`

Parameters for preview mode *AlongVhcl*

Sensor.Road.<no>.pos =x y z

Position of the sensor in vehicle frame (FrD). Coordinates x, y, z [m].

Example `Sensor.Road.<no>.pos = 4.0 0.0 0.5`

Sensor.Road.<no>.rot_z= *RotZ*

Specifies the rotation angle around the vehicle's z-axis of the preview direction relative to vehicle direction. Unit: [deg]. Default: 0.0.

Example `Sensor.Road.<no>.rot_z = 2`

Parameters for preview mode *OnlyRoadMarker*

Sensor.Road.<no>.SearchMarker = *MarkerString*

Specifies the road marker to be searched by the RoadSensor. Possible markers are:

`VelSign`	Search speed limit marker
`UserMarkerXXX`	Search any user defined marker

Default: `VelSign`

20.9.4 User Accessible Quantities

Please refer to section 24.14.7 'Road Sensor'.

20.10 Collision Sensor

20.10.1 Introduction

The Collision Sensor module detects if the ego vehicle body or wheels touch any traffic object. For the envelope of the vehicle body a cuboid or an extruded x-y contour can be defined. In general for the verification of collision the side areas of the traffic object and the vehicle body are considered. Thus it can be used for crash scenario or for parking maneuvers beside curbstone. Optionally using for the vehicle body a cuboid envelope a detection on upper and lower areas of the traffic object and vehicle body can be activated.

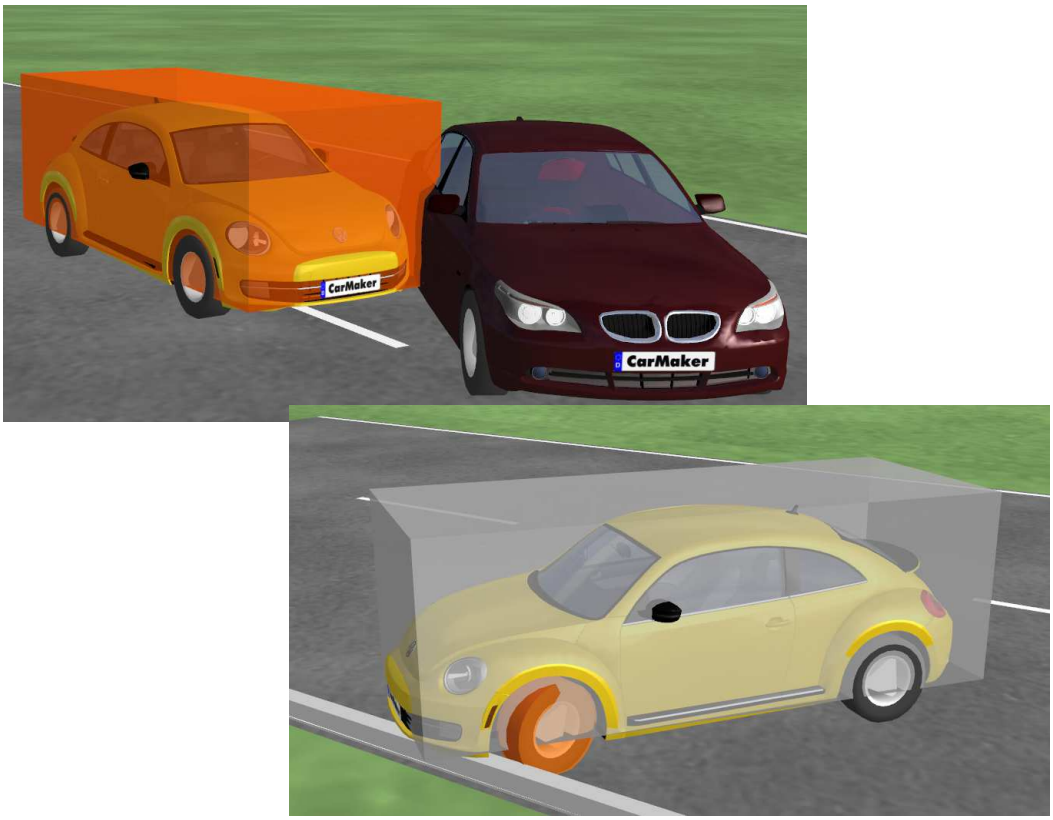


Figure 20.33: Collision Sensor with Vehicle Body and with Wheels

20.10.2 Parametrization of Collision Sensor

This collision sensor is parameterized in the vehicle file.

Sensor.Collision.Active

Activation of the collision sensor module. Default: 0.

Sensor.Collision.WithWheels

Specifies if the collision sensor for the vehicle wheels is active. Default: 0.

Sensor.Collision.Fr1.EnvMode

Specifies the envelope mode of the ego vehicle frame Fr1.

`Cuboid`	Vehicle envelope defined by vehicle outer skin as cuboid (Default)
`2DContour`	Vehicle envelope defined by extruded x,y-contour

Sensor.Collision.Fr1.AboveBelowOn

Specifies if traffic objects should be detectable on the upper and lower area and if the upper and lower ego vehicle areas from Fr1 can detect collision with traffic objects. This functionality is only possible for the vehicle envelope mode *Cuboid*. Default: 0.

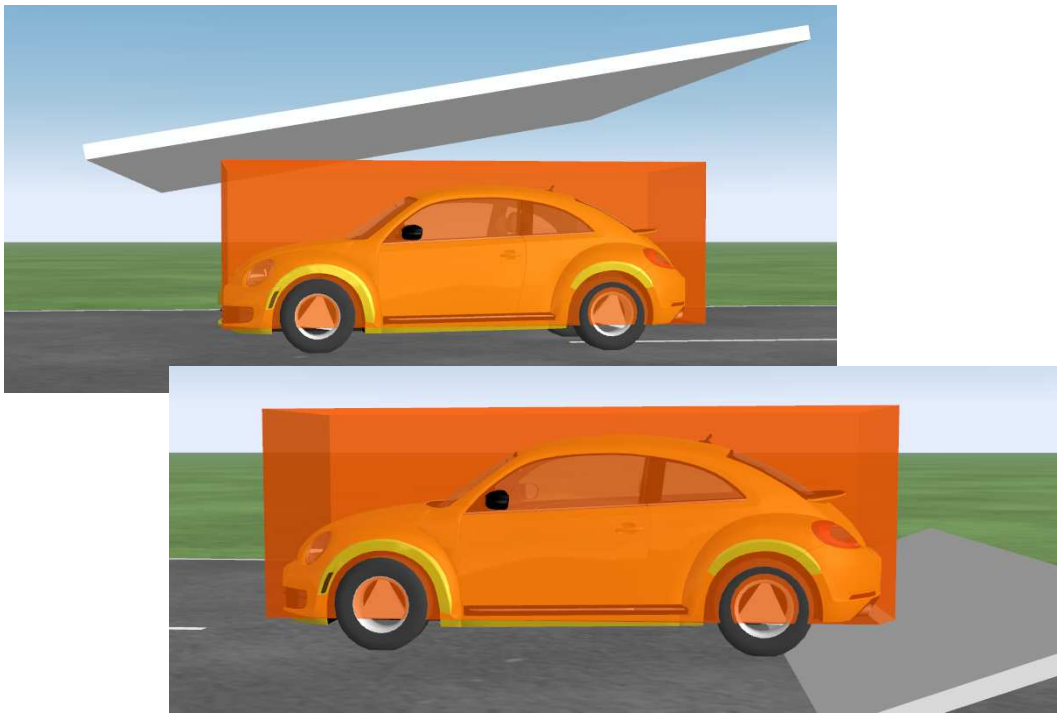


Figure 20.34: Collision Sensor on upper and lower traffic area

Parameters for the envelope mode 2DContour

Sensor.Collision.Fr1.Contour

Specifies the 2D lookup table for the vehicle Fr1 x-y contour [m]. The first point must start with 0/0. With active mirroring along the x-axis the y-value of the last value must be zero.

Example

```
Sensor.Collision.Fr1.Contour:
0.0 0.0
0.0 0.4
0.1 0.7
...
4.8 0.0
```

Sensor.Collision.Fr1.Contour.Mirror

Specifies if the x-y contour lookup table is mirrored along the vehicle Fr1 x-axis. Default: 1.

Sensor.Collision.Fr1.Contour.height

Specifies the height of the extruded 2D-contour [m]. The default value is the height from the vehicle outer skin.

Sensor.Collision.Fr1.Contour.xOff

Specifies the longitudinal offset between the Fr1 origin and the origin of the extruded 2D-contour [m]. The default value is the offset from the vehicle outer skin.

Sensor.Collision.Fr1.Contour.zOff

Specifies the vertical offset between the Fr1 origin and the bottom of the extruded 2D-contour [m]. The default value is the offset from the vehicle outer skin.

20.10.3 User Accessible Quantities

Please refer to section 24.14.8 'Collision Sensor' on page 801.

20.11 Radar Sensor

20.11.1 Introduction

The Radar Sensor generates a radar specific object list based on defined objects. It belongs to the class of HiFi Sensors and is designed especially for the needs of function development and testing.

The features of the Radar Sensor at a glance are:

- Detection based on signal-to-noise ratio (SNR) considering:
  - Detection threshold
  - Antenna characteristics
  - Object specific radar cross section (RCS)
  - Propagation / atmospheric losses
  - Object occlusion.
- Processing effects including:
  - Measurement noise
  - Latency due to processing time
  - Object merging due to lack of separability
  - False negatives
  - False positives.

The current chapter is structured as follows:

- Section 20.11.2 depicts the detection based on signal-to-noise ratio
- Section 20.11.3 illustrates the modeling of processing effects
- Section 20.11.4 outlines the output quantities
- Section 20.11.5 illuminates the parameterization.

20.11.2 Detection Based on Signal-to-Noise Ratio

The detection of the Radar Sensor is based on a physical model[1], if the signal-to-noise ratio SNR of an object is above a specific threshold:

$$SNR > SNR_{min}, \qquad \text{(EQ 453)}$$

the object will be detected. If the SNR is below the threshold, the detection will be discarded and a false negative will emerge.

Signal-to-Noise Ratio

The signal-to-noise ratio SNR is defined as quotient of the strength S of the received signal and the noise N:

$$SNR = \frac{S}{N}. \qquad \text{(EQ 454)}$$

1. The detection is explicitly not based on geometrical analyses. The parameters concerning range are merely used to identify detection candidates and to keep the simulation efficient. The parameters concerning field of view are used to calculate the antenna gain map.

Detection Threshold

The detection threshold is given by the minimal detectable value of the SNR SNR_{min} and results from the parameters minimal probability of detection P_{Dmin} and probability of false alarm P_{FA}:

$$SNR_{min} = 2(erfc^{-1}(2P_{FA}) - erfc^{-1}(2P_{Dmin}))^2 . \quad \text{(EQ 455)}$$

Signal Strength

The strength S of the received signal is determined by means of the radar equation. It considers the transmitted power P, the gain G of the antenna, the radar cross section RCS of the respective object, its distance r, the wave length λ as well as atmospheric L_{atm} and additional system L_A losses:

$$S = \frac{PG^2\lambda^2 RCS}{(4\pi)^3 r^4} \frac{1}{L_A L_{atm}} . \quad \text{(EQ 456)}$$

P	Transmitted power
G	Gain of the antenna $G_t = G_r$
λ	Wave length
RCS	Radar cross section
r	Distance to radar object
L_A	Additional system losses
L_{atm}	Atmospheric losses

Detection Noise

The influence of the signal chain on the noise N in the signal strength is incorporated by the detection noise. It considers the noise figure F_R, the noise bandwidth B_n as well as the environment temperature T_0:

$$N = T_0 F_R k_B B_n . \quad \text{(EQ 457)}$$

T_0	Temperature in K
F_R	Noise figure
k_B	Boltzmann constant
B_n	Noise bandwidth

Antenna Gain

The characteristics of the antenna are considered by means of a gain map. In the current release there is merely one map for both the transmitting and receiving antenna characteristics. The map can be created either by external tools, by measurements or by an internal model. The antenna gain map contains the one-way gain factor G for each direction. Its format is described in section 'Antenna Gain Map File'. The direction d is parametrized by azimuth φ and elevation θ in the sensor reference frame (cf. Figure 20.35). The boresight of the sensor coincides with the x-axis of its reference frame.

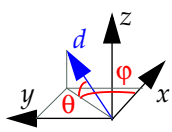


Figure 20.35: Direction by azimuth and elevation.

The internal model supports the common uniform rectangular aperture antenna. With the scan range parameters beam-steering can be simulated. Based on the beam width and height (3 dB one-way or 6 dB two-way) of the main lobe the aperture dimensions (a, b) are determined. Using the radiation pattern

$$f(\theta, \phi) = \frac{\sin(\pi \nu_y)}{(\pi \nu_y)} \frac{\sin(\pi \nu_z)}{(\pi \nu_z)} \quad \text{, with} \qquad \textbf{(EQ 458)}$$

$$\nu_y = \frac{a}{\lambda} \sin\theta \cos\phi \qquad \nu_z = \frac{b}{\lambda} \sin\theta \sin\phi \quad , \qquad \textbf{(EQ 459)}$$

field strength and gain of the antenna are calculated.

Figure 20.36 depicts an example of the horizontal and vertical sections of a antenna gain map generated with the internal model.

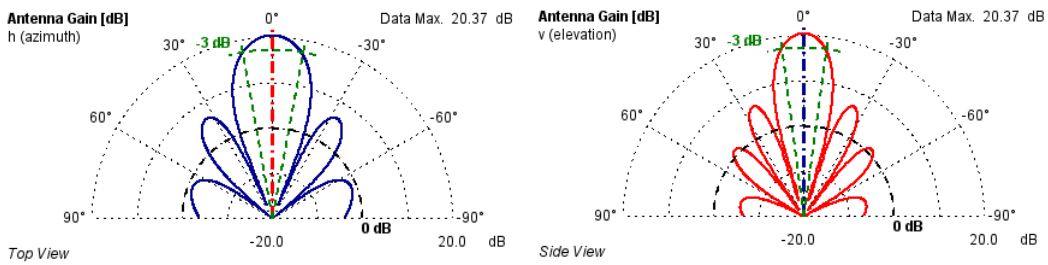


Figure 20.36: Antenna gain map horizontal and vertical sections.

Object Distance

The object's bounding box center BBC is selected as reference point for the following considerations (cf. Figure 20.37). Hence the distance of the object corresponds to the current distance between the bounding box center position and the sensor position.

Radar Cross Section

The radar cross section of an object depends on:

- Direction of incidence
- Statistical fluctuations
- Object occlusion and
- Object merging.

Direction of Incidence

The direction of incidence is measured with respect to the object's body fixed reference frame in the bounding box center BBC with the x-axis along the object's longitudinal axis and is parametrized by azimuth φ (cf. Figure 20.37).

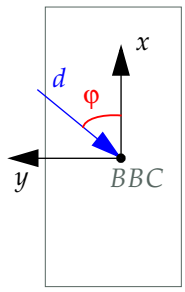


Figure 20.37: Direction of incidence by azimuth.

The azimuth dependency is reproduced by means of a measured look-up-table $RCS_{lut}(\varphi)$, the elevation dependency is modeled by the stochastic process (Swerling type 1, with Rayleigh-Distribution). (cf. Figure 20.38).

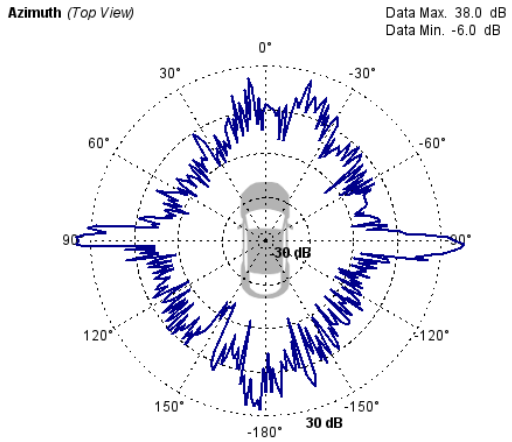


Figure 20.38: Radar cross section depending on direction of incidence.

An overview of all available pre-defined RCS maps can be found in section 'RCS map files'.

Statistical Fluctuation

In order to incorporate statistical fluctuations a Swerling type 1 model is used. The resulting RCS is distributed according to the probability density function:

$$P(RCS) = \frac{1}{cf(\theta)RCS_{lut}(\varphi)}\exp\left(\frac{RCS}{cf(\theta)RCS_{lut}(\varphi)}\right). \quad \textbf{(EQ 460)}$$

By default the stochastic part will cause a slightly different result for each testrun execution. In order to reproduce the same result please explicitly specify a seed value (GUI->Parameters->Additional Parameters: `RandomSeed = <value>`). See also section 2.3.2. If no seed value is specified here the default seed that is being used will be displayed in the session log.

Occlusion

Partial occlusion is considered based on the objects' bounding boxes. Using the bounding box corners relative horizontal o_h as well as vertical occlusion o_v (ranging from 0 to 1) is evaluated conservatively. The non-occluded RCS_{no} is modified according to:

$$RCS = (1 - o_h o_v)RCS_{no}. \quad \textbf{(EQ 461)}$$

Object Merging

In case of object merging due to lack of separability the RCS of the resulting object equals to the sum of the single values:

$$RCS = \sum_i RCS_i. \quad \textbf{(EQ 462)}$$

Damping

Atmospheric damping L_{atm} considers three different sources of power losses:

$$L_{atm} = L_{satm}L_{rain}L_{fog}. \quad \textbf{(EQ 463)}$$

Standard Atmospheric Damping

The losses in a standard atmosphere without rain or fog are described by the standard atmospheric damping. The used model $L_{satm}(f)$ depends on frequency.
[DESK, EW Class. Electronic Warfare and Radar Systems Engineering Handbook.]

Rain damping

Additional losses by rain are modeled based on an ITU recommendation. The implemented relation $L_{rain}(f, Q_{rain})$ depends on frequency and rain rate.
[ITU Recommendation P.838-3]

Fog damping

Additional losses by fog $L_{fog}(f, d_{vis})$ are modeled depending on frequency and visible distance.
[BROOKER, Graham. Introduction to sensors for ranging and imaging.]

20.11.3 Processing Effects

Measurement Noise

In order to model noisy measured data $\hat{x}$, the ground truth values x are enhanced by normally distributed random noise Δx:

$$\hat{x} = x + \Delta x \quad . \qquad \textbf{(EQ 464)}$$

The standard deviation of the noise is parametrized by the accuracy of the respective measurement. This results in a value $\hat{x}$ based on a confidence level of 1σ. You can achieve $N\sigma$ by dividing the accuracy by N.

By default the stochastic part will cause a slightly different result for each testrun execution. In order to reproduce the same result please explicitly specify a seed value (GUI->Parameters->Additional Parameters: `RandomSeed = <value>`). See also section 2.3.2. If no seed value is specified here the default seed that is being used will be displayed in the session log.

Time Delay

The Radar Sensor incorporates two different kinds of time delay, in order to depict the time progression during signal processing.

The Radar Sensor cycle time T_{radar} specifies the time difference between two calculations of the radar model in contrast to the simulation cycle time T_{sim}. During calculation the current data at this time is used.

The latency $\Delta t_{latency}$ specifies the time difference between calculation of the radar model and the update of the output (cf. Figure 20.39).

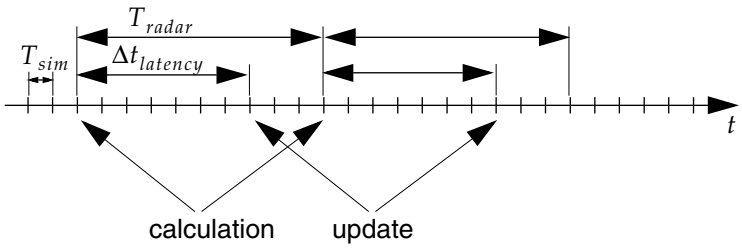


Figure 20.39: Influence of cycle time and latency

The time delay results in a age of output data between $\Delta t_{latency}$ and $\Delta t_{latency} + T_{radar}$.

Separability

The Radar Sensor has a finite resolution R in distance, velocity and azimuth. If two objects differ in all three states less than the respective separable value Δ_i:

$$\Delta_i = \delta R_i, \qquad \text{(EQ 465)}$$

the product of the separability δ and the respective resolution R_i, they are not differentiable anymore. As a consequence these objects are merged into a single object with a new bounding box enveloping the original ones. The output quantities are calculated for the new emerged object, the RCS corresponds to the sum of the original values.

False positives

Radar sensors sometimes show detections where there are no objects in reality. These detections are called “false positives”.

There are mirror objects, for example, when radar beams are reflected from the object via a guardrail or tunnel wall to the sensor. The probability of existence of a mirror object for trucks in our model is greater than that for cars. The probability of existence is depending on the distance from the sensor to the object and on the distance from the object to the guardrail / tunnel wall.

Traffic objects that are not placed on the road or the RoadMargin are not included in the calculation of the mirror objects.

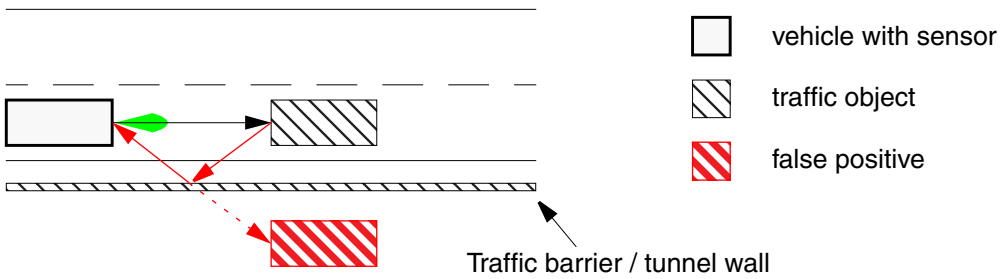


Figure 20.40: false positive

In addition to mirror objects, there are stochastically distributed false positives (clutter) due to signal noise. These are modeled using a Poisson distribution to alter the number of false positives and a uniform distribution to determine distance and object dimension quantities. The occurrence of false positives can be switched on and off via the GUI. One can also specify the expected number for the stochastically distributed false positives.

20.11.4 User Accessible Quantities

The Radar Sensor provides sensor instance specific (’Sensor Quantities’) as well as a object specific (’Object Quantities’) output quantities. The later ones are generally referred to as object list.

Objects keep their position in the object list as long as they are detected. Objects, which are detected for the first time, are sorted into free positions on the list. If the number of detected objects is larger than the list size N, only the closest objects to the sensor are sorted into the object list.

Sensor Quantities

The output quantities with respect to the sensor instance can be subdivided in quantities concerning:

- Detection
- Object administration
- Lateral position.

Detection

- Rolling counter
  Identification of detection update.
- Transmit power
  Modification of transmit power during simulation by DVA (e.g. depending on speed).

Object Administration

- Number of objects in detection
- Relevant target (number on object list)
  Selection of target for function algorithm.

Lateral Position

- Number of lanes left/right
- Distance to left/right border of the road.

Please refer to section 'General' on page 801.

Object Quantities

The quantities with respect to single objects can be subdivided in quantities referring to:

- Detection
- Kinematics
- Classification
- Ground truth information.

Detection

For the definition of the following quantities please refer to section 20.11.2.

- Radar cross section (RCS)
- Signal strength
- Signal-to-noise ratio (SNR)
- Probability of detection (resulting from (EQ 455)).

Kinematics

The Radar Sensor delivers the relative kinematics of the object's reference point, the bounding box center, with respect to the sensor frame regarding:

- Position
  Distance, longitudinal distance, lateral distance, vertical distance
- Velocity
  Relative speed, relative longitudinal speed, relative lateral speed
- Acceleration
  Relative longitudinal acceleration

- Orientation
  Relative course angle calculated from the measured velocity.

The ground truth data is enhanced by measurement noise (cf. section 20.11.3).

Classification

The following quantities allow for an evaluation of the object:

- Measurement status (no object, new object, object measured)
- Dynamic property (stationary, stopped, moving, oncoming)
- Object length/width (classified and floating point)
  Based on object definition and enhanced by measurement noise (cf. section 20.11.3)
- Probability of existence
  Object specific initialization after first detection. Will be incremented if object is detected, decremented otherwise. If zero, object will be dropped from object list. Therefore, an object can remain on the object list for several cycles, even if it is no longer detected. This integer value models the probability of existance. The following correlations may be assumed: 0 ~ 0%, 1 ~ 25%, 2 ~ 50%, 3 ~ 75%, 4 ~ 90%, 5 ~ 99%, 6 ~ 99.9%, 7 ~ 100%.
- Obstacle probability
  If an object is closer than the safety gap of 1 m to the driving path of the ego vehicle it is classified as obstacle candidate. The driving path is defined by the width of the ego vehicle, an additional clearance of 0.3 m and the current curvature radius of the trajectory. The obstacle probability is defined by $P_{obst} = 1 - g/(1m)$ using the gap g between the objects bounding box and the driving path (cf. Figure 20.41).

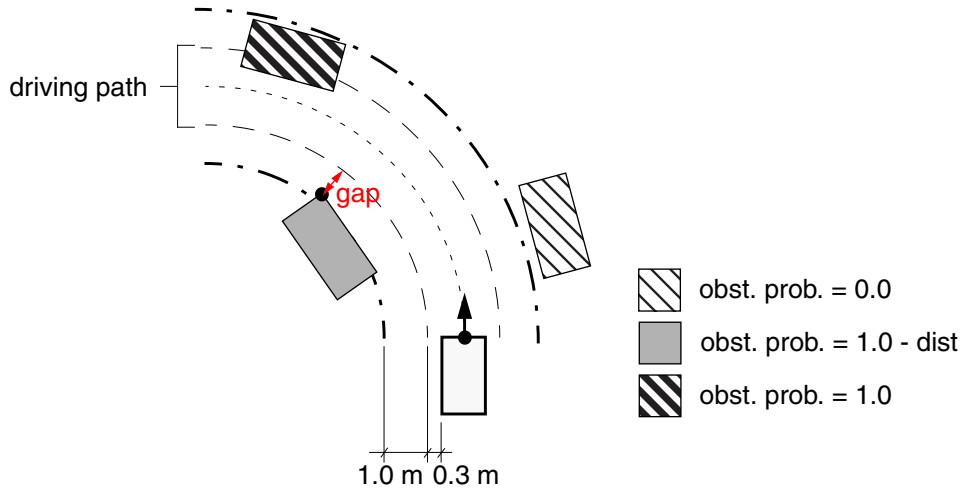


Figure 20.41: Obstacle probability by distance to driving path.

Ground Truth Information

- Global object ID
  Identification of a specified object.

Please refer to section 'Object List' on page 801.

20.11.5 Parametrization of Radar Sensor

The Radar Sensor is parametrized in the vehicle file. The signal-to-noise ratio is among others influenced by environmental conditions. These are described by the parameters temperature, rain rate as well as visual range in fog, which are specified in the environment module (see section 4.1 'General Parameters').

Sensor.Radar.N

Specifies the total number of radar sensors in the vehicle. Default: 0.

Sensor.Radar.<no>.name

A name for each sensor can be given.

Sensor.Radar.<no>.pos

Position of the sensor frame with respect to vehicle frame (FrD). Coordinates x, y, z [m].

Sensor.Radar.<no>.rot

Orientation of sensor frame with respect to vehicle frame (FrD). The x-axis of the sensor frame points in the direction of the sensor's boresight. Rotation angles rx, ry, rz [deg]. The rotation order is zyx.

Sensor.Radar.<no>.FoV

Field of view horizontal /vertical to limit the observation area [deg] (cf. Figure 20.42:). Default: 40.0 30.0

Sensor.Radar.<no>.MaxNumObj

Specifies the max. size of the object list (the max. no. of detectable objects). Default: 40.

Sensor.Radar.<no>.Range_max

This is the max. radius of the observation area [m] (cf. Figure 20.42). Default: 200 m.
To avoid the calculation of far-off objects, an observation area around the sensor can be defined. The sensor only checks the detection of objects within the observation area. Detection depends only on the calculated SNR value.
This parameter has huge impact on sensor performance!

Sensor.Radar.<no>.Range_min

Specifies the min. radius of the observation area [m] (cf. Figure 20.42). Default: 0.2 m.

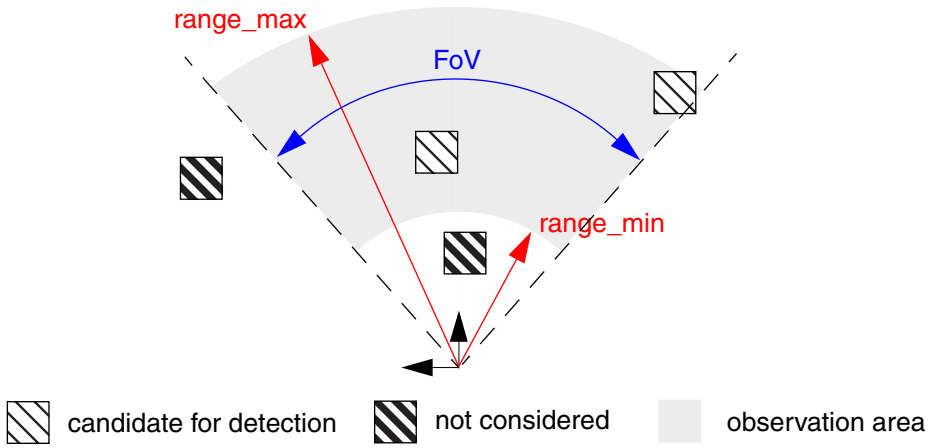


Figure 20.42: Observation area including candidates for detection.

Sensor.Radar.<no>.CycleTime

The CycleTime has to be given in [ms]. The value specifies the period of detecting the environment. Default: 60 ms. (cf. Figure 20.43)

Sensor.Radar.<no>.StartOffset

The StartOffset delays the calculation once at the beginning of the simulation. It is specified in [ms]. Default: 0 ms. (See Figure 20.43)

Sensor.Radar.<no>.Latency

Factor (0..1), that is multiplied with CycleTime, resulting in a time offset for updating the output quantities (cf. Figure 20.43). The resulting latency has to be less than the cycle time, so the factor specified in the file or the GUI has to be in the interval [0,1]. Default: 0.

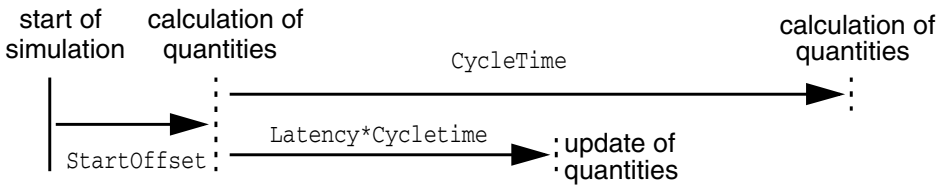


Figure 20.43: Timing of calculation and update of quantities.

Sensor.Radar.<no>.movietheme

Used for visualization of the sensor with *IPGMovie*. Possible values are:
`NotVisible`, `Cyan`, `Blue`, `Pink`, `Red`, `Yellow`, `Orange`, `Green`, `Magenta`

Sensor.Radar.<no>.Mounting

Indicates the reference frame on which the sensor is mounted. The vehicle body frames `Fr1A` and `Fr1B` are available. Default: `Fr1A`.

Sensor.Radar.<no>.Frequency

Specifies the transmit frequency of the sensor in [GHz]. Default: 77.0 GHz.
The RCS maps as well as the atmospheric damping model are merely validated for the frequency band around 77 GHZ.

Sensor.Radar.<no>.AccuracyDistance

Accuracy of distance measurement in [m]. Default: 0.4 m.

Sensor.Radar.<no>.AccuracyAzimuth

Accuracy of the azimuth measurement in [deg]. Default: 0.1 deg.

Sensor.Radar.<no>.AccuracySpeed

Accuracy of speed measurement in [km/h]. Default: 0.1 km/h.

Sensor.Radar.<no>.ResolutionDistance

Resolution of distance measurement in [m]. Default: 1.8 m.

Sensor.Radar.<no>.ResolutionAzimuth

Resolution of azimuth measurement in [deg]. Default: 1.6 deg.

Sensor.Radar.<no>.ResolutionSpeed

Resolution of speed measurement in [km/h]. Default: 0.4 km/h.

Sensor.Radar.<no>.Separability

Factor multiplied with resolution to get the minimal separable value [-]. Default: 1.5.

Sensor.Radar.<no>.TransmitPower

Transmit power of the sensor in [dBm]. Default: 14.0 dBm.

Sensor.Radar.<no>.SystemLosses

Additional losses in [dB]. Default: 0 dB.

Sensor.Radar.<no>.NoiseBandWidth

Noise bandwidth of the receiver in [Hz]. Default: 25000 Hz.

Sensor.Radar.<no>.NoiseFigure

Degradation of the signal-to-noise ratio, caused by components in the signal chain, for a given bandwidth in [dB]. Default: 4.8 dB.

Sensor.Radar.<no>.ProbDetectMin

Minimal probability of detection resulting in the threshold for detection [-]. Default: 0.5.

Sensor.Radar.<no>.ProbFalseAlarmIdx

Index for the probability of false alarm. The index values range from 1 to 10 and correspond to the probabilities from 10^{-1} to 10^{-10}. Default: 6.

Sensor.Radar.<no>.FalsePosActive

Specifies whether false positives are to be calculated or not. Default: 0.

Sensor.Radar.<no>.ClutterObjMean

Specifies the mean value of clutter objects in the observation area. Default: 5.

Sensor.Radar.<no>.GainFName

Specifies the name of the antenna gain map file. The file can be chosen/generated in the Sensor GUI. The file should be located in the “Data/Sensor” folder.

Antenna Gain Map File

The antenna gain data is read in via infofile as a 2D map depending on `Phi` (Azimuth) and `Theta` (Elevation). The user can generate antenna files depending on the two parameters `ScanRange h/v`, `BeamWidth h/v` and `Antenna Efficiency` (AzimutFoV / ElevationFoV) in the sensor GUI. The user can provide own files as well and select them in the Sensor GUI.

The following parameters have to be specified in the file:

FileIdent =	**CarMaker-AntennaGainMap**

ScanRange =	***value value***

Scan range h/v of antenna in [deg]. Use these two parameters to simulate beam steering. A resulting antenna gain map is calculated (see Figure 20.44).

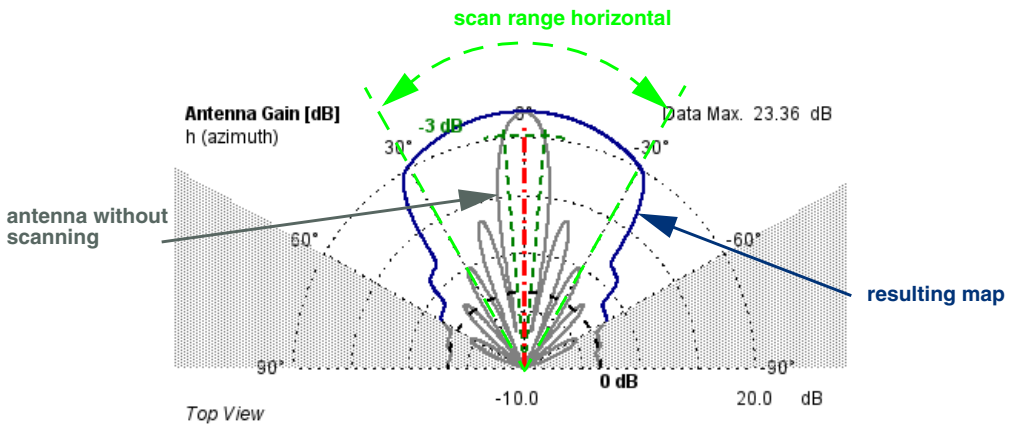


Figure 20.44: antenna with beam steering

BeamWidth =	***value value***

Beam width h/v of antenna in [deg] (-3 dB main lobe width, see Figure 20.45:). This parameter is shown in the GUI when loading a antenna file. The parameter is also used for the visualization of the antenna in IPGMovie.

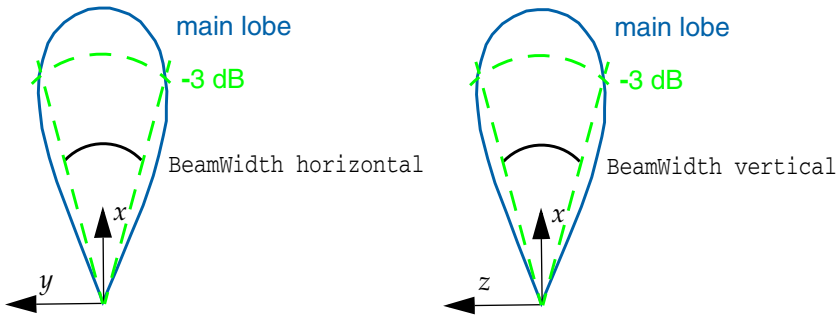


Figure 20.45: Definition of BeamWidth h/v

AntennaEff = ***value***

The antenna efficiency scales the antenna gain map during generation. This parameter is shown in the GUI when loading a file. Changing it in the file has no effect. If the parameter is modified in the GUI, a new file has to be created.

Type = **deg** or **rad**

Use “deg” or “rad”, depending on how `Azimuth` and `Elevation` are given.

Azimuth = ***values (min .. max)***

Specify **equidistant** sample points for the first coordinate from -90˚ degrees to 90˚ degrees (unit corresponding to `Type`). There should be at least five sample points. Use for example 1˚ steps and put the values in a single line.

Elevation = ***values(min .. max)***

Specify **equidistant** sample points for the second coordinate from -90˚ degrees to 90˚ degrees (unit corresponding to `Type`). There should be at least five sample points. Use for example 1˚ steps and put the values in a single line.

Gain: ***values (matrix: nAzimuth * nElevation)***

Specify gain values in a **n**x**m** matrix (n, m depending on the number of `Azimuth` and `Elevation` values). The antenna gain describes the ratio between the amount of energy propagated in a specific direction (`Azimuth`, `Elevation`) compared to the energy that would be propagated if the antenna were not directional.

RCS map files

Every object can be assigned to a RCS class in the Traffic GUI. CarMaker provides default RCS maps for the following classes:

RCS map	Approach
Pedestrian	Azimuth: Based on measurement data, simplified
Bicycle	Azimuth: Based on measurement data, simplified
Car	Azimuth: Based on measurement data, simplified
Truck	Azimuth: Based on RCS map of “Car”, extrapolated to fit to dimensions of a truck
Guardrail post	Azimuth: Based on PO simulations

The user can also choose the classes “User1”, “User2” or “User3”. In this case the appropriate RCS files have to be provided by the user. The file names have to be “RCS_User1”, “RCS_User2” or “RCS_User3”. The structure of the Infofile is described below. There is

also the possibility to “overwrite” the default files of the other classes. For that the user just needs to place his own file with corresponding name in the “Data/Sensor” folder of the project directory. Use the following filenames to overwrite the default files: “RCS_Pedestrian”, “RCS_Bicycle”, “RCS_Car”, “RCS_Truck”, “RCS_GuardRailPost”.

For some traffic objects currently no default RCS map is available. The RCS class for those objects is set to “RCS_Unknown” and the sensor can’t detect them.

Use the following parameters in the Infofile:

FileIdent =	**CarMaker-RCS Map**

Azim.type =	**deg** or **rad**

Use “deg” or “rad”, depending on how “Azim.data” is given.

Azim.data =	***values (min .. max)***

Specify **equidistant** sample points from -180˚ degrees to 180˚ degrees. There should be at least five sample points. Use for example 1˚ steps and put the values in a single line.

AzimRCS.type =	**linear, 10log10** or **20log10**

Use “linear”, “10log10” or “20log10”, depending on how “AzimRCS.data” is given.

AzimRCS.data =	***values***

Specify RCS value for each “Azim.data” sample point in a single line.

20.12 Global Navigation Sensor

20.12.1 Introduction

The Global Navigation Sensor simulates the GPS (Global Positioning System) satellites and their visibility for the receiver on the vehicle.

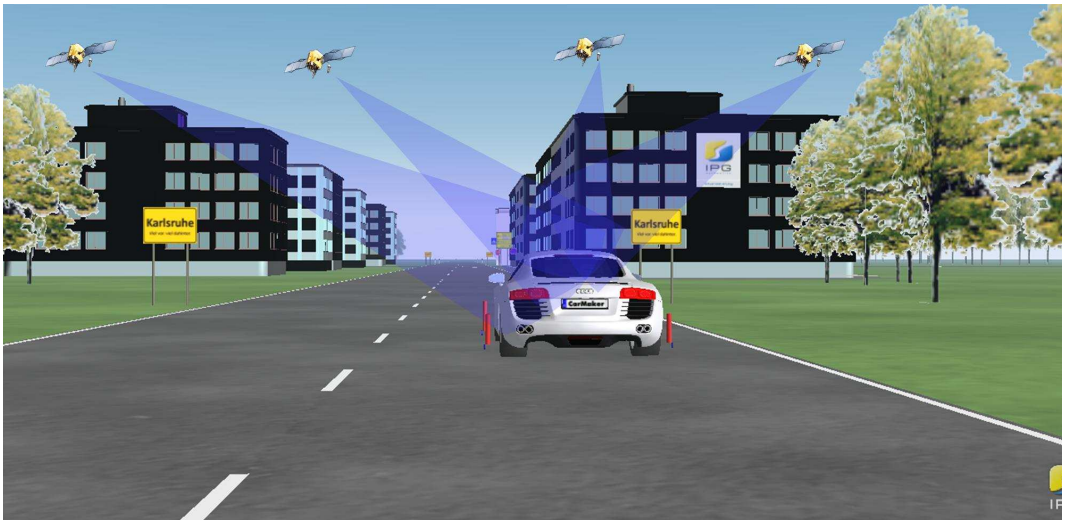


Figure 20.46: Global Navigation Sensor

This sensor module calculates the position and velocity of all GPS satellites in their orbits using the navigation message, a file containing all parameters to simulate the position of the satellites for a given time. With the use of error models the pseudorange between the satellite and the vehicle is calculated. Traffic objects can be considered to only use the satellites with a clear line of sight for calculating the position of the receiver.

The Position is given as accurate position, position from pseudorange and delta position - the difference between the former. The used coordinate systems are the cartesian coordinates (x,y,z) Earth Centered Earth Fixes (ECEF) and the ellipsoidal coordinates (φ, λ, h), where φ describes the latitude, λ the longitude and h the height (elevation) above the ellipsoid. See also section 'Geographic Coordinate System (GCS)' on page 733.

20.12.2 Calculation

In reality the pseudorange is obtained by correlating the satellites Pseudo-Random Noise (PRN) signal with a local copie to get the transmission time and from that the pseudorange to the satellites. In CarMaker the PRN code is not simulated and neither are the different frequences. Instead the position of the satellites and the receiver in ECEF coordinates is calculated and used to get the geometrical range. By adding delays due to different error sources the pseudorange is obtained. All error sources, except of the satellite clock error, can be turned on and off by the user. Correction data for the satellite clock error are included in the GPS navigation message.

To simulate the satellites the navigation message files for the given date and time have to be available for CarMaker. The files can be downloaded from ftp://cddis.gsfc.nasa.gov/pub/gps/data/daily. Select the year of interest and go to the "brdc" folder to download the required files. Only the files ending with "n.Z" are required. These are the files for the GPS Satellites, the files ending with "g.Z" contain the data for the GLONASS Satellites and are not supported by CarMaker. The files have to be placed in the CarMaker project directory

under "Data/Sensor/EphemRinex/rinex_xxxx", where "xxxx" stands for the corresponding year. A total of 32 satellites can be simulated with CarMaker. If more satellites are operating the ones with PRN higher than 32 will not be simulated.

The position of all GPS satellites is calculated with ephemeris parameters given in the navigation message. The exact position of the receiver is given from CarMaker and can be used to calculate the geometric distance between the receiver and the satellites. Adding delays due to different errors will resulte in the so called pseudorange. To calculate the receiver position as well as the receiver clock error at least four satellites signals have to be visible to the receiver at any given time. With at least four signals the four unknows can be calculated using least square iteration. CarMaker uses all-in-view calculation, which means all satellites, which are visible to the receiver, will be used for the calculation.

The geometric dilution of precision (GDOP) is calculated from the exact position of the receiver and the position of the satellites. The GDOP is a dimensionless parameter, which describes the accuracy of the calculated receiver position and changes for different satellite constellations. The lower the GDOP value, the better the calculation of the receiver position. If the GDOP is higher than 20, no position will be calculated in CarMaker, even if there are four or more satellites in view.

The user can define an elevation mask. Satellites that are below the given elevation value won't be visible for the receiver. An elevation mask is applied because satellites with a low elevation value experience higher ionospheric and tropospheric delays, due to a longer propagation time through the atmospheric layers.

20.12.3 Error Modelling in CarMaker

The different error sources as well as their modelling in CarMaker will be discussed here. Most of the errors are modelled as a first order Gauss-Markov process and can be parametrized with the correlation time and the standard derivation.

Satellite Clock Error

The navigation message contains parameters to calculate the satellite clock error. Normally the parameters are corrected once a day. The clock is simulated with broadcasted parameters. This error can not be turned off in CarMaker.

Ephemeris Error

The GPS satellite positions can differ from their orbits specified in the navigation message. The orbits are controlled and the navigation message is updated usually every two hours to account for the errors. Due to the ephemeris error data from the navigation message shouldn't be used, if they are older than four hours. The error from ephemeris data is modelled as a first order Gauss-Markov process.

Ionospheric delay

The ionosphere is an atmospheric layer ranging from 50 to1000km. It contains a large number of ionized particles, which leads to a delay of the signal. In CarMaker, it can be modelled as a first order Gauss-Markov process or with the Klobuchar Model. If the Klobuchar Model is used the delay has no effects on the position from pseudorange, because the calculated error will be used as the sole ionospheric error and corrected in the calculation of the position from pseudorange.

Tropospheric delay

The troposhere consists of the lower part of the earth's atmosphere. In CarMaker the troposheric delay can be modelled as a first order Gauss-Markov process or with an Ionomodel, the Modified Tropospheric Refraction Correction Model. If the latter is used the delay has

no effects on the position from pseudorange, because the calculated error will be used as the sole tropospheric error and corrected in the calculation of the position from pseudorange.

Multipath Error

This error is not modelled in CarMaker.
Multipathing exists due to reflection of the signal off the surrounding. The reflected signal will be measured with a higher pseudorange because of the longer propagation time. Because of the complexity and dependency on the frequency of the signal among other factors this error is not simulated in CarMaker.

Receiver Clock Error

The receiver clock is not synchronized with the satellite clocks and not as accurate. To account for the receiver clock error a total of four satellites have to be in view. That way the position and the receiver clock error can be calculated. This error is modelled as a first order Gauss-Markov process

Receiver Noise

The receiver noise is modelled as random gaussian and can be parameterized with the standard derivation of the pseudorange and standard derivation of the rangerate.

20.12.4 Visibility Constrains due to Traffic

The visibility of a satellite can be obscured due to buildings and other traffic objects. To consider these the "Calculate DirectView" option has to be activated. Using this option, the line of sight to the satellite is used to check if a traffic object is in the way. Only satellites with a direct view to the receiver will be used for calculation of the user position and the geometric dilution of precision (GDOP).

20.12.5 Parametrization of the Global Navigation Sensor

The calculation of the satellites is activated in the environment module, as well as the parameterization of the date and time, see section 4.1 'General Parameters' on page 49.

The Global Navigation Sensor is parameterized in the vehicle file.

Sensor.GNav.pos

Position of the sensor in vehicle frame (FrD). Coordinates x, y, z [m].

Example `Sensor.GNav.pos = 2.0 0.0 1.5`

Sensor.GNav.Mounting

Indicates the frame on which the sensor is mounted. There are the two vehicle body frames Fr1A and Fr1B available. Default: Fr1A.

Example `Sensor.GNav.Mounting = Fr1B`

Sensor.GNav.UpdRate

Specifies with which rate the sensor detects the environment and updates the signals [Hz]. Default: 10.

Sensor.GNav.nCycleOffset

Specifies the number of cycles after which the sensor begins with the calculation of the signals. Default: 0.

Sensor.GNav.DirectView

Specifies if traffic objects should be considered. Only satellites with an unobscured view will be considered for the calculation. Default: 0.

Sensor.GNav.ShowDDict

Specifies if Rinex data should be included as UAQs. Please note this will add over 700 entries to the dictionary. Default: 0.

Sensor.GNav.ElevationMask

Specifies the elevation mask of the receiver [deg]. Satellites with an elevation below the elevation mask will not be used for calculation. Default: 10.

Sensor.GNav.ReceiverClockError.Active

If set to 1 the receiver clock error is simulated. Default: 0.

Sensor.GNav.ReceiverClockError.std

Specifies the standard deviation for the first order gauss-markov formula to simulate the receiver clock error [m]. Default: 15.0.

Sensor.GNav.ReceiverClockError.corrTime

Specifies the correlation time for the first order gauss-markov formula to simulate the receiver clock error [ms]. Default: 3600.

Sensor.GNav.CommonMode.Active

If set to 1 the common mode errors are simulated. This includes the ephemeris error as well as the atmospheric effects from the ionosphere and the troposphere. Default: 0.

Sensor.GNav.EphError.std

Specifies the standard deviation for the first order gauss-markov formula to simulate the ephemeris error [m]. Default: 3.0.

Sensor.GNav.EphError.corrTime

Specifies the correlation time for the first order gauss-markov formula to simulate the ephemeris error [ms]. Default: 1800.

Sensor.GNav.IonoModel.Active

If set to 1 the ionospheric error is modelled using the Klobuchar model. Default: 0.

Sensor.GNav.IonoError.std

Specifies the standard deviation for the first order gauss-markov formula to simulate the ionosperic error [m]. Default: 5.0.

Sensor.GNav.IonoError.corrTime

Specifies the correlation time for the first order gauss-markov formula to simulate the ionospheric error [ms]. Default: 1800.

Sensor.GNav.TropoModel.Active

If set to 1 the troposperic error is modelled using the Modified Tropospheric Refraction Correction Model. Default: 0.

Sensor.GNav.TropoError.std

Specifies the standard deviation for the first order gauss-markov formula to simulate the tropospheric error [m]. Default: 2.0.

Sensor.GNav.TropoError.corrTime

Specifies the correlation time for the first order gauss-markov formula to simulate the tropospheric error [ms]. Default: 3600.

Sensor.GNav.ReceiverNoise.Active

If set to 1 the receiver noise error is simulated. Default: 0.

Sensor.GNav.ReceiverNoise.stdPsr

Specifies the standard deviation for the pseudorange noise [m]. Default: 0.1.

Sensor.GNav.ReceiverNoise.stdRr

Specifies the standard deviation for the range rate noise [m]. Default: 0.05.

20.12.6 User Accessible Quantities

Please refer to section 24.14.10 'Global Navigation Sensor' on page 803.

20.13 Ultrasonic Raw Signal Interface

20.13.1 Introduction

The Ultrasonic Raw Signal Interface (RSI) simulates the propagation of mechanical sound pressure waves through the virtual scenario using ray tracing. For each echo it provides information such as time of flight and sound pressure amplitude. The Ultrasonic RSI can be used either to reconstruct an analog signal as input for the processing unit or as complete sensor component with the built-in simple processing.

The features of the Ultrasonic RSI at a glance are:

- Included effects:
  - Direct echo
  - Indirect echo
  - Repeated echo
  - Cross echo
  - Road clutter
  - False positives
  - False negatives
- Wave propagation considering:
  - Detailed 3D geometry
  - Reflection
  - Scattering
- Sound pressure amplitude incorporating:
  - Propagation losses
  - Transducer directivity
  - Atmospheric absorption
  - Reflectivity
  - Coherent addition
- Processing including:
  - Echo separability
  - Time dependent threshold

The Ultrasonic RSI can be applied in three different modes:

- Reconstruction of analog raw signal using impulse response as input for user specific coherent addition and detection algorithm.
- Detection by sound pressure amplitude considering coherent addition and time dependent threshold.
- Detection by 3D field of view considering wave propagation.

The current chapter is structured as follows:

- Section 20.13.2 illustrates the included effects
- Section 20.13.3 describes the modeling of wave propagation
- Section 20.13.4 introduces the processing
- Section 20.13.5 sketches object model requirements
- Section 20.13.6 outlines the output quantities
- Section 20.13.7 illuminates the parameterization.

20.13.2 Included Effects

The physical modeling of sound pressure wave propagation (cf. section 20.13.3) enables inherently the consideration of the major effects in ultrasonic transducer arrays for automotive parking applications. A classification of different propagation paths is introduced in the following.

Direct Echo

If a sound pressure wave is reflected only once, for instance by another car's bumper, and received by the transmitting transducer it will be called direct echo (cf. Figure 20.47). The direct echo constitutes the shortest propagation path compared to the followings.

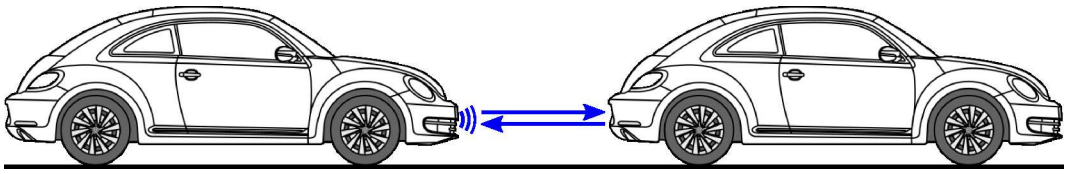


Figure 20.47: Direct echo by another car's bumper

Indirect Echo

If the sound pressure wave is reflected at least twice, for instance by another car's bumper and the road, and received by the transmitting transducer it is referred to as indirect echo (cf. Figure 20.48). In case a corresponding direct echo exists, the indirect echo results in a second echo with a slightly longer propagation path.

Strictly speaking, the echo of a channeling built up by a curb and the road is also an indirect echo as it involves more than one reflection (cf. Figure 20.49).

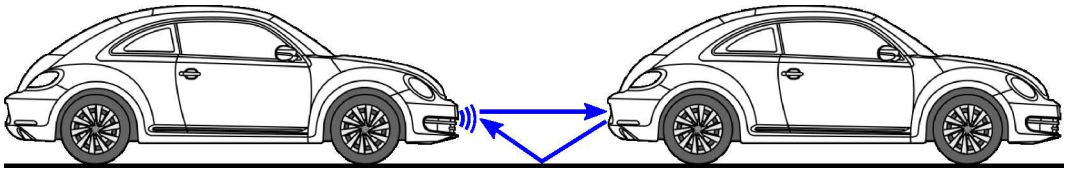


Figure 20.48: Indirect Echo by another car's bumper and the road

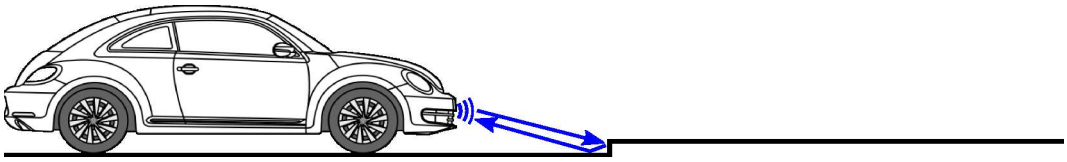


Figure 20.49: Indirect echo by a curb

Repeated Echo

If the sound pressure wave is also reflected by the own car's bumper, the propagation path can be passed back and forth multiple times, which is called a repeated echo (cf. Figure 20.50). The length of the repeated echo corresponds to a multiple of the respective direct echo and results in a second, third, ... echo.

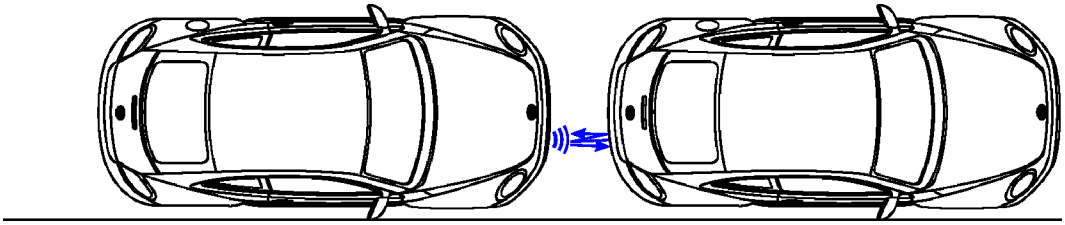


Figure 20.50: Repeated echo by another car's and the own car's bumpers

Cross Echo

If the reflected sound pressure wave is received by a transducer different to the transmitting one, a cross echo will be detected (cf. Figure 20.51). Cross echoes are useful to distinguish the form of an obstacle, for instance a pollard and a wall and to determine its orientation.

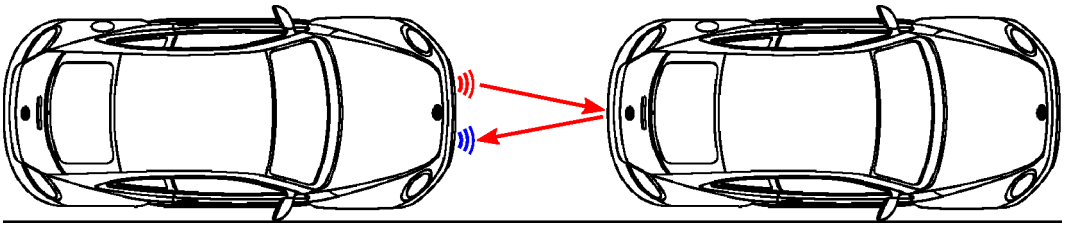


Figure 20.51: Cross echo of by another car's bumper

Road Clutter

If the sound pressure wave is reflected directly by the road without further interactions, road clutter will emerge (cf. Figure 20.52).

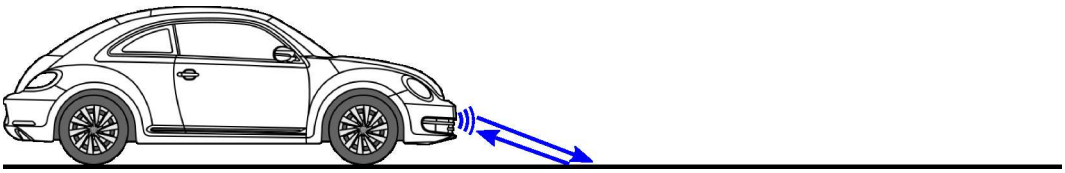


Figure 20.52: Road clutter

False Positives

The main source of false positives is road clutter. An indirect or repeated echo additional to a direct echo usually does not lead to critical false positives, as these echoes have a longer propagation path. In contrast indirect or repeated echoes are used as additional information and for plausibility checks.

False Negatives

If the reflected sound pressure drops below a specified threshold, the respective obstacle will not be detected any more and a false negative will emerge. One reason could be the form and relative orientation of the reflecting surface. If the obstacle lacks corner reflectors

or roundings and merely consists of inclined planes with sharp edges, the object will reflect the sound pressure away from the transducer (cf. Figure 20.53). Further causes for low reflected sound pressure will be introduced in section 20.13.3.

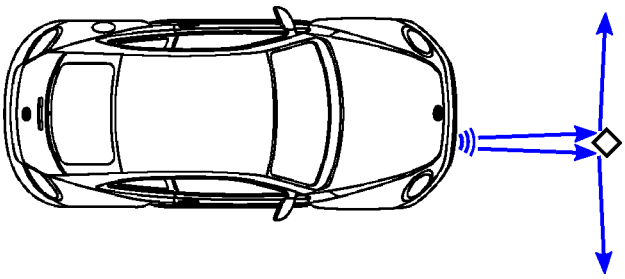


Figure 20.53: Stealth pollard

20.13.3 Wave Propagation

A ray tracing algorithm is applied to the virtual scene, to simulate the propagation of the sound pressure wave.

Propagation Geometry

Transmission

The transducer transmits the sound pressure wave in the region specified as field of view. The wavefront is discretized by rays, the interaction with the scenario is determined for each ray separately.

Reflection and Scattering

The search of the reflection point as well as the calculation of the reflection direction is based on the detailed 3D geometry of the visualization. In order to account for scattering effects of rough surfaces a material dependent stochastic component is introduced.

By default the stochastic part will cause a slightly different result for each testrun execution. In order to reproduce identical result over several testrun executions please explicitly specify a constant seed value.

Receiving

A ray can be received by a transducer if its absolute deviation is below a threshold called sensitivity. Receiving only states the necessary condition for detection, the sufficient condition is depicted in section 20.13.4.

Sound Pressure Amplitude

The sound pressure amplitude considers the propagation loss, transducer directivity, atmospheric absorbtion, reflectivity and optionally coherent addition.

Propagation Loss

The decay of the sound pressure amplitude caused by the spread of the wave front is described by the propagation loss.

Transducer Directivity

The spatial extension of the ultrasonic transducer's diaphragm results in a sound pressure distribution or directivity and is identical for transmission and receiving.

Atmospheric Absorption

During the travel through the atmosphere, attenuation of the sound pressure wave due to absorption occurs. The decay of the sound pressure amplitude is determined according to ISO 9613 and depends on the frequency as well as atmospheric conditions consisting of air temperature, air pressure and air humidity.

Reflectivity

The decay of the sound pressure amplitude by reflection is considered dependent on the material of the reflecting surface.

Coherent Addition

The transducer emits an extended wave package specified by the transmission time and sound propagation speed depending on temperature. If two or more wave packages overlap during receiving they will interfere with each other.

In case of frequency modulated transmission the assumed phase relation no longer holds and the coherent addition should be toggled off by setting the transmission time to zero. In this case the user has to take care about the correct interference of the signals.

20.13.4 Processing

Echo Separability

If two echoes from the same transmitter have a similar time of flight, they can not be separated anymore due to wave package overlap or a not decayed diaphragm of the transducer. To account for this effect, echoes with a difference in time of flight smaller than the specified separability margin are merged and only the nearest echo is reported.

Time Dependent Threshold

The detection is based on a time of flight dependent threshold for the sound pressure amplitude in order to counteract the distance dependent decay and to balance the detection of false positives and false negatives.

In order to disable the processing set the separability margin as well as the threshold for the sound pressure amplitude to zero.

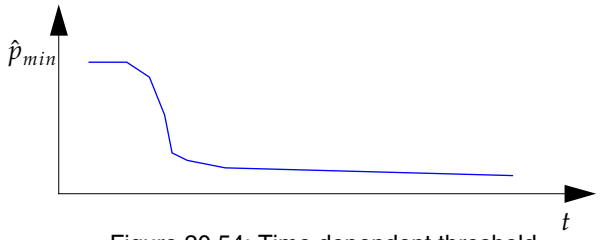


Figure 20.54: Time dependent threshold

3D Field of View

A detection based on the 3D field of view is provided alternatively. For each received ray it is checked whether the half length of flight is below the specified maximum range depending on the transmission direction.

20.13.5 Object Model Requirements

The Ultrasonic RSI calculates the interaction of the sound pressure wave with virtual obstacles. In order to get a meaningful result, the object models have to fulfill certain requirements:

- The faces have to be small enough to represent the curvature and to contain relevant corner reflectors. If uneven parts like a radiator grill are modeled by textures (photo wallpaper) they will not act as corner reflectors.
- The normals have to point perpendicularly outwards of a closed surface. If normals do not deliver what their name promises, the Ultrasonic RSI will get wrong inputs and draw wrong conclusions.
- The faces have to be tagged with a suitable material (cf. section 4.3 'Material Parameters').

In general the object model library delivered with CarMaker fulfills these requirements. Publicly available object models usually do not.

Currently there are two exceptions concerning the use of CarMakers object models:

- Animated characters as people or animals are approximated by their bounding box.
- Trees are approximated by cylinders representing their trunk.

20.13.6 User Accessible Quantities

The Ultrasonic RSI provides the following output quantities for each transducer:

- Number of echoes
- Echo
  - Time of flight
  - Length of flight
  - Sound pressure amplitude
  - Sound pressure level
  - Number of reflections
  - Transmitter to identify cross echo.

For details, please refer to section 24.14.11 'Ultrasonic RSI' on page 805.

20.13.7 Parametrization of the Ultrasonic RSI

The Ultrasonic RSI is parametrized in the vehicle file.

Atmospheric absorbtion and sound propagation speed are influenced by atmospheric conditions. These are described by the parameters air temperature, air pressure and air humidity, which are specified in the environment module (cf. section 4.1 'General Parameters').

Scattering as well as reflectivity are material dependent and specified in the material library (cf. section 4.3 'Material Parameters').

To get deterministic behavior, disable the stochastic part by specifying a random seed (cf. section 2.3.2 'Random number generator').

Class parameters

Sensor.USonicRSI.N = int

Total number of ultrasonic transducers built in the vehicle. Default: 0.

Sensor.USonicRSI.CycleTime = int

The cycle time [ms] specifies the period of detecting the environment and updating the respective quantities. All transducers of the Ultrasonic RSI are calculated at a time. The user is in charge to omit information of not active transducers. Default: 10.

Sensor.USonicRSI.Latency = double double

Relative min. and max. latency multiplied with cycle time results in a range for quantity update (cf. Figure 20.55). A physical sensor has a latency close to cycle time. This results in a parameter value of 1.0. Default: 1.0 1.0.

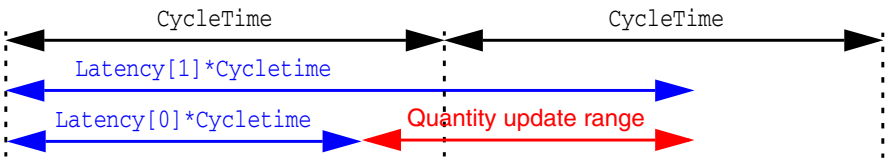


Figure 20.55: Quantity update range.

This parameter has huge impact on sensor performance!
The latency value is used internally to handle synchronization of CPU and GPU calculation. If the two values differ, the synchronization is weak, the simulation faster, but jitter will occur. Different synchronization modes can be adjusted:

- Quantity update without jitter [1.0 1.0]
- Quantity update with weak synchronization, allowing for faster simulation [1.0 2.0]
- Quantity update as soon as possible [0.0 1.0]

The enforcement of synchronization is handled differently depending on the simulation type. If the calculation does not finish in time, the simulation will be slowed down in CarMaker/Office whereas a realtime violation occurs in CarMaker/HIL. New sensor calculation is started after quantity update.

Sensor.USonicRSI.Reflections = int

The maximum number of reflections considered. Default: 4.

Sensor.USonicRSI.Sensitivity = double

The sensitivity represents the maximum deviation of a received ray from the sensor [deg]. Default: 1.0.

Sensor.USonicRSI.Detection = SPA / 3DFoV

In mode `SPA` the detection is determined based on the sound pressure amplitude, whereas mode `3DFoV` is based on a 3D field of view considering wave propagation. Default: `SPA`.

Sensor.USonicRSI.Visualization = bool

Toggles visualization of all sensors at once. Default: `0`.

Sensor.USonicRSI.Scattering = bool

Toggles scattering of all sensors and all materials at one time. Default: `0`.

Sensor.USonicRSI.Quantities = bool

Enables creating quantities. For a hugh number of segments disabling is highly recommended, especially in realtime simulations. Default: 0.

Instance Parameters

Sensor.USonicRSI.<no>.name = string

Name in form of an unique identifier. Default: USRS00.

Sensor.USonicRSI.<no>.CrossEchoes = bool .. bool

Array of size `Sensor.USonicRSI.N` specifying potential transmitters of received echoes (`CrossEchoes[<no>]=1`). Default: 0 .. 0 1 0 .. 0 (no cross echoes).

Sensor.USonicRSI.<no>.pos = double double double

Position of the transducer frame with respect to vehicle frame (FrD). Coordinates x, y, z [m]. Default: 0.0 0.0 0.0.

Sensor.USonicRSI.<no>.rot = double double double

Orientation of transducer frame with respect to vehicle frame (FrD). The x-axis of the transducer frame points in the direction of the transducer's boresight. Rotation angles rx, ry, rz [deg]. The rotation order is zyx. Default: 0.0 0.0 0.0.

Sensor.USonicRSI.<no>.FoV = double double

Field of view of the transducer specified by horizontal and vertical opening angle [deg]. The vertical field of view should not exceed 80 deg. Default: 120.0 60.0.

Sensor.USonicRSI.<no>.Range = double double

Range of the transducer specified by minimal and maximal distance [m]. The range heavily influences the efficiency of the calculation, by determining the propagation area. Default: 0.1 6.0.

Sensor.USonicRSI.<no>.nRays = int int

Discretization of field of view by vertical number of rays. Horizontal number of rays results from field of view and the condition of equidistant angles. If the error message out of memory occurs, reducing the number of rays might help. Default: 259 150.

Sensor.USonicRSI.<no>.nEchoesMax = int

Maximum number of echoes limits the output. If more echoes exist, the echoes with longer time of flight will be omitted. Default: 20.

Sensor.USonicRSI.<no>.Frequency = double

The frequency [kHz] is used to calculate the coefficient of atmospheric absorbtion, speed of sound and the relative phase for coherent addition. Default: 50.0.

Sensor.USonicRSI.<no>.TransTime = double

The transmission time [ms] is taken into account for coherent addition to determine the wave package overlap of echoes from an identical transmitter. Default: 0.3.

Sensor.USonicRSI.<no>.EchoSepa = double

Required difference in time of flight [ms] to separate two echoes from an identical transmitter. Should be greater or equal to transmission time. Default: 0.4.

Sensor.USonicRSI.<no>.Mounting = Fr1A / Fr1B

Indicates the reference frame on which the transducer is mounted. The vehicle body frames `Fr1A` and `Fr1B` are available. Default: `Fr1A`.

Sensor.USonicRSI.<no>.VisColor = NotVisible / Red / ...

Color used for visualization of the transducer in *IPGMovie*. Default: NotVisible.

Sensor.USonicRSI.<no>.SLobeFName = string

Name of the sound lobe map file. It can be selected/generated in the GUI. The file should be located in the folder `/Data/Sensor`. Default: `USonicRSI_default`.

Sensor.USonicRSI.<no>.ThresSPA: double double

2D array of sound pressure amplitude detection threshold [Pa] as function of time [ms].

Sound Lobe Map File

The sound lobe is specified in a separate file located in the folder `/Data/Sensor`.

Sound lobe maps generated by external FE Tools can be used providing the information specified in section 'Required Parameters for Simulation'.

CarMaker comes with a generator of sound lobe maps which needs additional information depicted in section 'Optional Parameters for Generation'.

Required Parameters for Simulation

FileIdent = CarMaker-SoundLobeMap

SoundLobe.Type = SPA / 3DFoV

In a testrun the sound lobe map `Type` has to fit to the `Detection` mode.

SoundLobe.FoV = double double

Dimensions of the sound lobe definition specified by the horizontal and vertical field of view [deg].

SoundLobe.DeltaFoV = double double

Sampling period of the horizontal and vertical opening angle [deg] of the sound lobe.

SoundLobe.SPA0Range = double

Reference range [m] at which the sound pressure amplitude is given.

SoundLobe.Map:
double double ... double

Map of sound pressure amplitude at reference range `SPA` [Pa] values or maximum range `RangeMax` [m] values , depending on `SoundLobe.Type`.

It is assumed that the sample points of the 2D sound lobe map correspond to directions d parametrized by equidistant azimuth α and elevation β coordinates (cf. Figure 20.56).

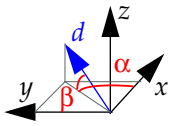


Figure 20.56: Spherical coordinates by azimuth and elevation.

The sound lobe map data is ordered starting with the biggest values for azimuth $\alpha = \eta/2$ and elevation $\beta = \nu/2$ emerging from the half of the opening angles η and ν. Within one row the azimuth decays with the specified sampling period $\Delta\eta$, within one column the elevation with the specified sampling period $\Delta\nu$ (cf. (EQ 466)). This corresponds to a image from transducer point of view.

$$\begin{array}{ccccc} \beta\backslash\alpha & \frac{\eta}{2} & \frac{\eta}{2}-\Delta\eta & \dots & -\frac{\eta}{2} \\ \frac{\nu}{2} & a_{0,0} & a_{0,1} & a_{0,j} & a_{0,m-1} \\ \frac{\nu}{2}-\Delta\nu & a_{1,0} & a_{1,1} & a_{1,j} & a_{1,m-1} \\ \dots & a_{i,0} & a_{i,1} & a_{i,j} & a_{i,m-1} \\ -\frac{\nu}{2} & a_{n-1,0} & a_{n-1,1} & a_{n-1,j} & a_{n-1,m-1} \end{array} \quad \text{(EQ 466)}$$

Optional Parameters for Generation

SoundLobe.Roundness = double

Grade of roundness of the sound lobe generated from the corresponding lookup table. Values smaller than one result in convex shapes. Default: 1.0.

SoundLobe.Ratio = double

Ratio of vertical and horizontal dimensions of the sound lobe generated from the corresponding lookup table. Default: 0.5.

SoundLobe.RangeMax:
double double

Array of sound pressure amplitude [Pa] at reference range as function of horizontal angle [deg] for the positive horizontal field of view (required for `SoundLobe.Type 3DFoV`). The sampling points of the horizontal angle start at zero and have to be equidistant.

SoundLobe.SPA0:
double double

Array of maximum range [m] as function of horizontal angle [deg] for the positive horizontal field of view (required for `SoundLobe.Type SPA`). The sampling points of the horizontal angle start at zero and have to be equidistant.
.

20.14 Radar Raw Signal Interface

20.14.1 Introduction

The Radar Raw Signal Interface (RSI) simulates the propagation of electromagnetic waves through the virtual scenario using ray tracing. For each channel it provides information such as time of flight, relative electric field strength amplitude and relative doppler shift. The Radar RSI is not a complete sensor model, it merely delivers the channel impulse response, sufficient information to reconstruct an analog signal.

The features of the Radar RSI at a glance are:

- Included effects:
  - Multipath propagation
  - Repeated path propagation
  - Doppler shift
  - Road clutter
  - False positives
  - False negatives
- Wave propagation considering:
  - Detailed 3D geometry
  - Reflection
  - Scattering
- Relative electric field strength amplitude incorporating:
  - Propagation losses
  - Atmospheric absorption
  - Material and polarization dependent reflection

The current chapter is structured as follows:

- Section 20.14.2 illustrates the included effects
- Section 20.14.3 describes the modeling of wave propagation
- Section 20.14.4 sketches object model requirements
- Section 20.14.5 outlines the output quantities
- Section 20.14.6 illuminates the parameterization.

20.14.2 Included Effects

The physical modeling of electromagnetic wave propagation (cf. section 20.14.3) enables inherently the consideration of the major effects in radar sensors for automated driving applications. The considered effects are not limited to the below mentioned ones, these are merely examples for the sake of illustration.

Multipath Propagation

If an electromagnetic wave is reflected by more than one target on its way from transmitter to receiver the necessary condition for multipath propagation will be fulfilled. Usually a direct channel exists where the electromagnetic wave takes the straight path from transmitter to target and back to receiver. In most scenarios road vehicles encounter further propagation channels.

Typical road surfaces are perceived by electromagnetic millimeter waves as mirrors. So one additional channel consists of a path of electromagnetic waves via the road (cf. Figure 20.57). The direct channel and the channel via the road differ in length and so their incoming electromagnetic waves in phase. The different phases result in interference effects, which lead to signal strength increasing or decreasing and jitter.

Figure 20.57: Multipath propagation via road

On highways metallic guardrails provide another mirroring surface (cf. Figure 20.58). So another additional channel consists of a path of electromagnetic waves via the guardrail. In this case also interference effects can occur. The radar sensors capability of determining the horizontal direction of a target leads furthermore to gusting. An additional ghost target occurs as mirror image of the real target.

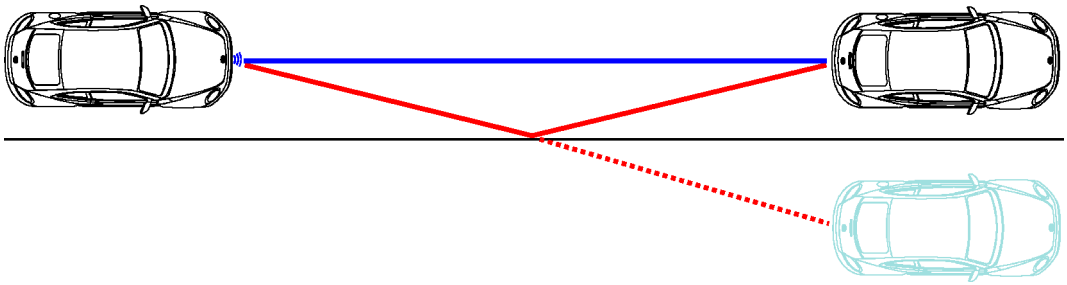


Figure 20.58: Multipath propagation via guardrail

Multipath propagation is not limited to road and guardrail, another infamous example would be a drive through a tunnel.

Repeated Path Propagation

If the detected target is sufficiently close to the transceiver repeated path propagation will occur. The incoming electromagnetic wave is reflected by the structure of the ego vehicle surrounding the transceiver and travels the way between target and transceiver back and forth twice. The radar sensor has no information to identify this effect and unfolds the channel to a channel of double length. This results in an additional ghost target.

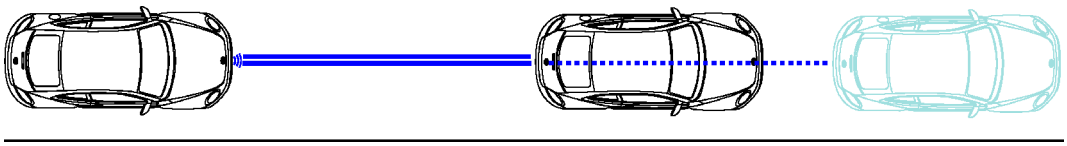


Figure 20.59: Repeated path propagation with double travelled way

Doppler Shift

If the electromagnetic wave is reflected by a target with a speed relative to the transmitter or last reflection point, doppler shift appears. The RadarRSI delivers the resulting frequency shift incorporating all reflections relative to the transmitter's frequency. The radar sensor can use this information in combination with its own speed in order to determine the dynamic status of the target as standing, oncoming or departing.

Road Clutter

If an electromagnetic wave is directly reflected by the road without further interactions, road clutter will emerge (cf. Figure 20.60).

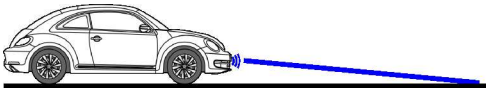


Figure 20.60: Road clutter

False Positives

False positives originate in the above mentioned effects multipath propagation, repeated path propagation, doppler shift and road clutter. Multipath and repeated path propagation deliver ghost objects. The differences in doppler shift of a vibrating driver's cab and the rest of a truck result in two separated targets. The direct reflection of the road delivers clutter.

False Negatives

There are various reasons for false negatives based on signal strength of a single channel which will be introduced in section 20.14.3. Negative interference in multipath propagation will additionally lower the signal strength and can lead to false negatives. Another mentionable contribution originates from the form and relative orientation of the reflecting surface. If the target lacks corner reflectors or roundings and merely consists of inclined planes with sharp edges, it will get stealth characteristics by reflecting the electromagnetic wave away from the transceiver (cf. Figure 20.61) .

Figure 20.61: Stealth target

20.14.3 Wave Propagation

A ray tracing algorithm is applied to the virtual scene, to simulate the propagation of the electromagnetic wave.

Propagation Geometry

Transmission

The transceiver transmits the electromagnetic wave in the region specified as field of view. The wavefront is discretized by rays, the interaction with the scenario is determined for each ray separately.

Reflection and Scattering

The search of the reflection point as well as the calculation of the reflection direction is based on the detailed 3D geometry of the visualization. In order to account for scattering effects of rough surfaces a material dependent stochastic component is introduced.

Scattering centers, which are not modeled with detailed geometry can be tagged with a special scatter material[1]. Faces which are tagged as scatterers reflect the ray back into the direction of incidence with a small stochastic perturbation.

By default the stochastic part will cause slightly different results for each testrun execution. In order to reproduce identical results over several testrun executions, please explicitly specify a constant seed value.

Receiving

A ray will be received if its relative distance is below a threshold depending on the ray sampling period and the receivers sensitivity. Receiving only states the necessary condition for detection, the sufficient condition has to be checked by the user implementation.

Relative Electric Field Strength

The relative electric field strength amplitude considers the propagation loss, atmospheric damping and polarization dependent reflectivity.

Propagation Loss

Decay of the electric field strength caused by spreading of the wave front is described by propagation loss.

Atmospheric Damping

Travelling through the atmosphere, damping of the electric field strength occurs due to standard atmosphere, rain and fog. Damping of the electric field amplitude is determined according to section 20.11.2 and depends on frequency, rain rate and visual range in fog.

Reflectivity

Decay of the electric field strength by reflection considers the direction as well as relative polarization of the incident electromagnetic wave as well as the relative permittivity of the material by means of the Fresnel coefficients.

20.14.4 Object Model Requirements

The Radar RSI calculates the interaction of the electromagnetic wave with virtual targets. In order to get a meaningful result, the object models have to fulfill certain requirements:

- The faces have to be small enough to represent the curvature and to contain relevant corner reflectors. If uneven parts like a radiator grill are modeled by textures (photo wallpaper) they will not act as corner reflectors.
- The normals have to point perpendicularly outwards of a closed surface. If normals do not deliver what their name promises, the Radar RSI will get wrong inputs and draw wrong conclusions.
- The faces have to be tagged with a suitable material (cf. section 4.3 'Material Parameters').

In general publicly available object models do not fulfill these requirements.

Currently there are two limitations concerning the use of object models:

- Animated characters such as people or animals are approximated by their bounding box.
- Trees are approximated by cylinders representing their trunk.

1. Tagging faces as scatterer will only make sense, if the geometry model omits the cause for the scattering center, in most cases corner reflectors. For instance in a head lamp the metal parabolic reflector behind the glass or plastic pane is mostly omitted. To get the reflection anyway the effect is modeled by tagging.

20.14.5 User Accessible Quantities

The Radar RSI provides the following output quantities for each transceiver:

- Number of channels
- Channel
  - Time of flight
  - Length of flight
  - Relative electric field strength amplitude depending on polarization
  - Relative doppler shift
  - Number of reflections
  - Direction of transmitted and received electromagnetic wave

For details please refer to section 24.14.12 'Radar RSI' on page 805.

20.14.6 Parametrization of the Radar RSI

The Radar RSI is parametrized in the vehicle file.

Atmospheric damping is influenced by atmospheric conditions. These are described by the parameters rain rate and visual range in fog specified in the environment module (cf. section 4.1 'General Parameters').

Scattering as well as relative permittivity are material dependent and specified in the material library (cf. section 4.3 'Material Parameters').

To get deterministic behavior, disable the stochastic part by specifying a constant random seed (cf. section 2.3.2 'Random number generator').

Class Parameters

Sensor.RadarRSI.N =	**int**

Total number of Radar transceivers built in the vehicle. Default: 0.

Sensor.RadarRSI.CycleTime =	**int**

The cycle time [ms] specifies the period of detecting the environment and updating the respective quantities. All transceivers of the Radar RSI are calculated at a time. The user is in charge to omit information of not active transceivers. Default: 60.

Sensor.RadarRSI.Latency = double double

Relative min. and max. latency multiplied with cycle time results in a range for quantity update (cf. Figure 20.62). A physical sensor has a latency close to cycle time. This results in a parameter value of 1.0. Default: 1.0 1.0.

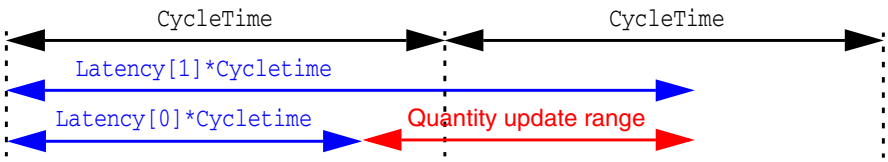


Figure 20.62: Quantity update range.

This parameter has huge impact on sensor performance!
The latency value is used internally to handle synchronization of CPU and GPU calculation. If the two values differ, the synchronization is weak, the simulation faster, but jitter will occur. Different synchronization modes can be adjusted:

- Quantity update without jitter [1.0 1.0]
- Quantity update with weak synchronization, allowing for faster simulation [1.0 2.0]
- Quantity update as soon as possible [0.0 1.0]

The enforcement of synchronization is handled differently depending on the simulation type. If the calculation does not finish in time, the simulation will be slowed down in CarMaker/Office whereas a realtime violation occurs in CarMaker/HIL. New sensor calculation is started after quantity update.

Sensor.RadarRSI.Reflections = int

The maximum number of reflections considered. Default: 3.

Sensor.RadarRSI.Sensitivity = double

Sensitivity of receiving the wavefront. With a value of 1.0 only rays perfectly hitting the transceiver are received. Default: 10.0.

Sensor.RadarRSI.Visualization = bool

Toggles visualization of all sensors at one time. Default: `0`.

Sensor.RadarRSI.Scattering = bool

Toggles scattering of all sensors and all materials at once. Default: `0`.

Sensor.RadarRSI.Quantities = bool

Enables creating quantities. For a hugh number of segments disabling is highly recommended, especially in realtime simulations. Default: 0.

Instance Parameters

Sensor.RadarRSI.<no>.name = string

Name in form of an unique identifier. Default: RARS00.

Sensor.RadarRSI.<no>.pos = double double double

Position of the transceiver frame with respect to vehicle frame (FrD). Coordinates x, y, z [m]. Default: 0.0 0.0 0.0.

Sensor.RadarRSI.<no>.rot = double double double

Orientation of transceiver frame with respect to vehicle frame (FrD). The x-axis of the transceiver frame points in the direction of the transceiver's boresight. Rotation angles rx, ry, rz [deg]. The rotation order is zyx. Default: 0.0 0.0 0.0.

Sensor.RadarRSI.<no>.FoV = double double

Field of view of the transceiver specified by horizontal and vertical opening angle [deg]. Default: 120.0 60.0.

Sensor.RadarRSI.<no>.Range = double double

Range of the transceiver specified by minimal and maximal distance [m]. The range heavily influences the efficiency of the calculation, by determining the propagation area. Default: 0.1 70.0.

Sensor.RadarRSI.<no>.nRays = int int

Discretization of field of view by vertical number of rays. Horizontal number of rays results from field of view and the condition of equidistant angles. If the error message out of memory occurs, reducing the number of rays might help. Default: 1211 700.

Sensor.RadarRSI.<no>.nChansMax = int

Maximum number of channels limits the output. If more channels exist, the channel with longer time of flight will be omitted. Default: 160.

Sensor.RadarRSI.<no>.Frequency = double

The frequency [GHz] is used to calculate the coefficient of atmospheric damping as well as the absolute doppler shift. Default: 77.0.

Sensor.RadarRSI.<no>.ChanSepa = double

Required difference in length of flight [m] to separate two channels from an identical transmitter. If two channels are not separable, the one with the longer path is omitted. Default: 0.4.

Sensor.RadarRSI.<no>.Mounting = Fr1A / Fr1B

Indicates the reference frame on which the transceiver is mounted. The vehicle body frames `Fr1A` and `Fr1B` are available. Default: `Fr1A`.

Sensor.RadarRSI.<no>.VisColor = NotVisible / Red / ...

Color used for visualization of the transceiver in *IPGMovie*. Default: NotVisible.

Chapter 21
Traffic

21.1 Overview

The task of the traffic module is to define additional road users and to simulate common traffic situations. Possible applications are ADAS, EURONCAP, Autonomous Driving, etc.

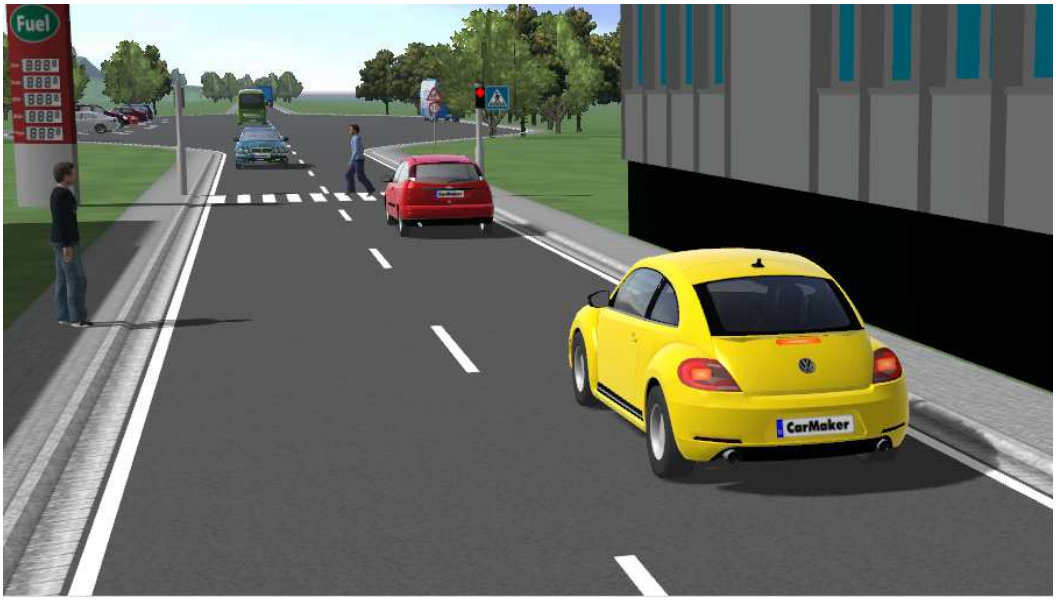


Figure 21.1: City traffic

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.
- In motion or stationary.
- Performing closed loop and open loop driving maneuvers independently from other road users.
- Existing for a given period of time.

- Different motion models:
  - Single track model for 4-wheel and 2-wheel objects
  - Pedestrian model
  - Ball motion model
- Motion attributes:
  - maximal velocity,
  - maximal longitudinal acceleration and deceleration,
  - maximal lateral acceleration.

Predefined traffic object classes with a set of basic and vehicle dynamic parameters are available.

Figure 21.2: Traffic on motorway

Figure 21.4: Child playing with a ball on the road

21.2 Interface

Interface for the initialization function

The following table represents the input and output variables of the initialization interface struct *tTrafficObj_Cfg*:

Variable	Unit	Description
Id		Id number of the traffic object
RouteId		Id number of the route taken by the traffic object
ObjectKind		Traffic object kind (moveable, stationary)
MotionKind		Traffic object motion kind (4-wheel, 2-wheel, course)
Name		Traffic object name
Info		Traffic object info
DetectMask		Detection mask: first bit: detectable by sensors second bit: detectable by other traffic objects, driver
w, h, l	m	Width, height, length
zOff	m	Vertical offset
Envelope*		Object envelope areas information
attrib		Additional traffic object attributes
RCSClass		Radar cross section class for Radar Sensor
r_zyx[3]	rad	Euler angles z-y-x for FrRoad->FrTraffic
sRoad	m	Traffic object initial road coordinate on route, path
tRoad	m	Traffic object initial lateral distance to route, path
v	m/s	Traffic object initial velocity

Interface for the evaluation function

The following table represents the input and output variables of the evaluation interface struct *tTrafficObj*:

Variable	Unit	Description
State		State of the object: 0: motion deactivated, object hidden 1: motion activated 2: static object (e.g.: building) 3: user defined motion
DetectLevel		Detection level of the object by sensor: 0: not detected 1: detected, not crucial 2: detected, crucial
sRoad	m	Road coordinate on route, path
tRoad	m	Lateral distance to selected route or path
t2Ref	m	Lateral distance to road reference line
Distance	m	Driven distance
LongVel	m/s	Longitudinal velocity in course orientation

Variable	Unit	Description
LongAcc	m/s^2	Longitudinal acceleration in course orientation
LatVel	m/s	Lateral velocity relative to route, path y axis (derivation of tRoad)
r_zyx[3]	rad	Rotation angles of traffic object
Tr2Fr0[3][3]		Transformation matrix from traffic object frame to inertial frame Fr0
t_0[3]	m	Global position of the traffic object
t_GCS		GCS Coordinates (Latitude, Longitude, Elevation)
v_0[3]	m/s	Velocity of the traffic object in global frame
a_0[3]	m/s^2	Acceleration of the traffic object in global frame
v_1[3]	m/s	Global velocity in traffic object frame
a_1[3]	m/s^2	Global acceleration in traffic object frame
rv_zyx[3]	rad/s	Derivation of Euler angles z-y-x
omega_0[3]	rad/s	Rotational velocity (Fr0)
Information about position on road network		
LinkObjId		Current road link object Id
LaneObjId		Current road lane object Id
onJunction		Flag if traffic object on road junction
JuncObjId		Current (last) junction object Id
nextJuncObjId		Next road junction object Id
s2nextJunc	m	Road distance along road reference line to next junction
s2lastJunc	m	Road distance along road reference line to last junction
<preLane> := Lane.<X>.* (X:= Act, OnLeft, OnRight) - information about the current driving lane and the left/right neighboring lane		
<preLane> .LaneId		Lane Id
<preLane> .isRight		Lane left/right in direction of the route
<preLane> .Type		Lane typ (e.g. driving lane, side-strip, ...)
<preLane> .Width	m	Lane width
<preLane> .tMidLane	m	deviation between the middle of the lane and the reference line/route
Information about the object envelope for sensors		
Envelope*		Object envelope areas information in global frame

21.3 General Parameters

The parameters explained here apply for all traffic objects and are stored in the TestRun infofile.

Traffic.N

Specifies the total number of traffic objects. Default: 0.

Traffic.SpeedUnit

Selects the global speed unit: kmh or ms.

Example `Traffic.SpeedUnit = kmh`

Traffic Object Parameters

A prefix is built up for each traffic object by the string “Traffic.“, followed by the Traffic number <Id> starting with zero for the first traffic object. The keys of all parameters for each traffic object start with this prefix.

In the traffic dialog there are some predefined settings consisting of a movie geometry file and the corresponding infofile (with the extension **.objinfo*) with all required traffic object parameters, which are imported automatically by loading the movie geometry file:

Predefined Setting	Movie Geometry
Compact Car	3D/Vehicles/VW_Beetle_2012_Blue.mobj
Sports Car	3D/Vehicles/Porsche_911Turbo_2012.mobj
Rigid Truck	3D/Vehicles/MB_Actros_8x4_Dumper.mobj
Semi Truck	3D/Vehicles/MB_Actros_1996+Trailer.obj
Bus	3D/Vehicles/MB_CitaroO345_2005.mobj
Coach	3D/Vehicles/Coach.mobj
Motorcycle	3D/Vehicles/Suzuki_Demo_Traffic_Driver.obj
Bicycle	3D/People/Biker_Female01.manim
Pedestrian	3D/People/Pedestrian_Male_Casual01_IPG.manim
Animal	3D/Animals/Moose.manim
Ball	3D/StreetFurniture/Ball_Soccer.mobj

Traffic.<Id>.ObjectKind = *string*

Selection of the traffic object kind. The traffic library provides the following object kinds:

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

Example `Traffic.1.ObjectKind = Movable`

Traffic.<Id>.Name = *name*

A name for each object (except object kind *StatNoName*) can be given. The User Accessible Quantities related to a single traffic object include its name string: Traffic.<Name>.Parameter . Default: T00.

Traffic.<Id>.ObjectClass = *class*

Specifies the traffic object class. Following classes are supported:

- Unknown (default)
- Toy
- Animal
  People
- Bicycle
  Motorcycle
- Car
- Van
  Caravan
- Bus
  Coach
- Truck
  Truck_Semi

Example `Traffic.0.ObjectClass = Car`

The object class is used by the autonomous driver and for positioning of an static object.

Traffic.<Id>.Info = *string*

Description with additional information about the traffic object.

Example `Traffic.1.Info = oncoming traffic, color: dark red`

Traffic.<Id>.Movie.Geometry = *FileName*

File name to the movie geometry file used to identify the object.

Example `Traffic.1.Movie.Geometry = Coach.mobj`

Traffic.<Id>.Color = *R* *G* *B*

Box Color, defined in RGB.

Example `Traffic.1.Color = 1.0 0.65 0.0`

Traffic.<Id>.Basics.Dimension = *l* *w* *h*

These parameters define the overall dimensions of a traffic object [m].

Example `Traffic.2.Basics.Dimension = 4.8 1.8 1.2`

Traffic.<Id>.Basics.Offset = *zOff* *lOff*

The z-offset defines the distance of the traffic object from the road surface.
The l-offset is currently not supported.

Traffic.<Id>.Basics.Fr12CoM = *value*

Specifies the x coordinate of the center of mass (CoM) [m]. Default: length/2.

Traffic.<Id>.Attrib = *string*

Assign up to 10 attributes to a traffic object. All attributes need to be declared in the c-code interface. The single attribute items are separated by a blank.

Traffic.<Id>.RCSClass = *class*

Specifies the radar cross section class for the Radar Sensor. Following classes are supported:

- RCS_Unknown (default)
- RCS_Pedestrian
- RCS_Bicycle
- RCS_Motorcycle
- RCS_Car
- RCS_Truck
- RCS_Building
- RCS_User1
  RCS_User2
  RCS_User3

Example `Traffic.0.RCSClass = RCS_Car`

Traffic.<Id>.DetectMask = *bool <Sensor> bool <AutonomousDriving>*

If the parameter is set to 1, the traffic object is detectable for Object Sensors and Free Space Sensors of the test car and/or for other traffic objects with autonomous driving. Default: 1.

Example `Traffic.2.DetectMask = 1 1`

2D contour

The 2D contour allows the presentation of symmetric rounded objects.

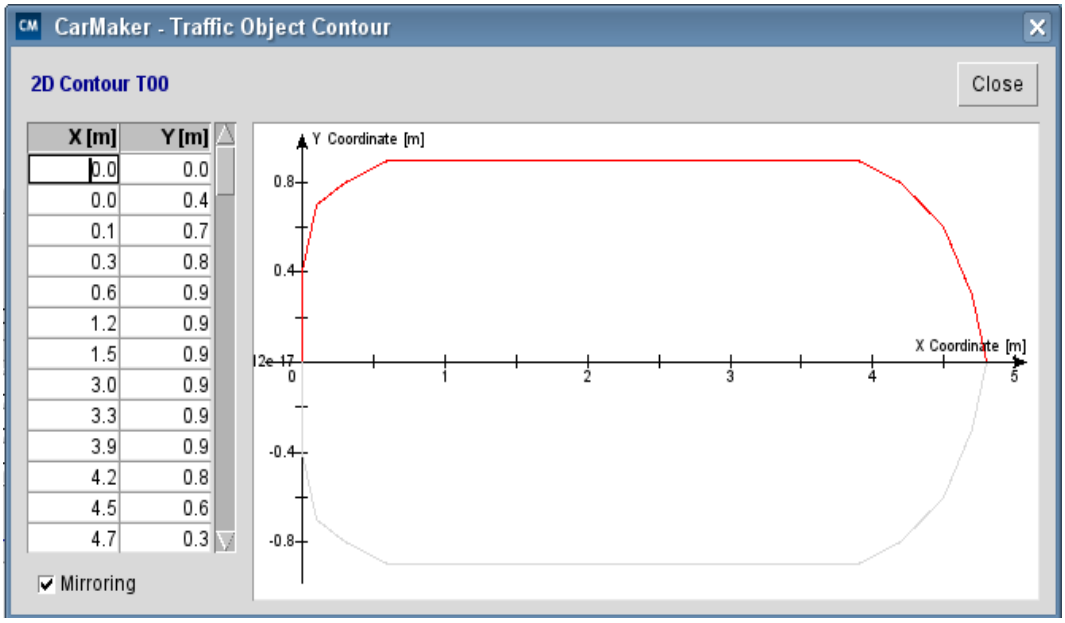


Figure 21.6: 2D contour

Traffic.<Id>.Basics.Contour.Mirror = *bool*

Parameter for mirroring one side of the traffic object. Default: 1.

Traffic.<Id>.Basics.Contour: *DataTable*

Using a contour of the traffic objects instead of the rectangular box.

Syntax — Infofile look-up table with 2 columns
<x-coordinate [m]> <y-coordinate [m]>

Example

```
Traffic.3.Basics.Contour:
0.0 0.0
0.1 0.7
...
```

Starting Conditions

In case a road network is available, each traffic object can follow an individual path. For oncoming traffic the reverse path of the ego vehicle can be used.

It is possible to place/move an object on a route path (using the parameter **.Route*) or on a dynamic path (using the parameter **.DynPath*):

Traffic.<Id>.Route = ***<Route: Id or Name> bool <UsePath>***

Selects the route path (with fixed path sequence) to use. The route can be selected by route Id (road internal index 0..N) or route name. If 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 route (reference line path). Default: 1 (use the path).

Example `Traffic.0.Route = 0 1` or `Traffic.0.Route = Route_A2B 1`

Traffic.<Id>.DynPath = ***<List of path object Ids: ObjId0 ... ObjId4 >***

Selects the initial dynamic path (path sequence can be changed) to use: one or up to five paths (lane, user or connector path) specified by their global road obejct Id=*ObjIdx*, which can be changed or followed up in maneuver. In the case of more than one, each next path must be a valid successor.

Example
```
Traffic.0.DynPath = 154
Traffic.0.DynPath = 154 10 55 95 12
```

Traffic.<Id>.Init.Orientation = X Y Z

Specifies the initial orientation of the traffic object on the road referred to the route/path orientation [deg]. The rotation sequence is z-y-x. Default: 0.0 0.0 0.0.

Traffic.<Id>.Init.v = ***value***

Specifies the start velocity of the traffic object [Traffic.SpeedUnit]. Default: 0.

Traffic.<Id>.Init.Road = ***sRoad tRoad*** **or**
Traffic.<Id>.Init.Road = ***sRoad absoluteLane***

Start position of the traffic object in road coordinates. The coordinates refer to the object's rear plane (traffic frame origin).

<sRoad> = longitudinal direction along route or path [m]
<tRoad> = lateral deviation to route or path [m]
<absoluteLane> = specifies the driving lane (e.g. L0, R1).

Example `Traffic.0.Init.Road = 150 L0`

21.4 Motion model

Following motion model parameters are only used for a moveable traffic object.

Traffic.<Id>.Motion.Kind = ***KindStr***

Specifies the motion kind of the traffic object. Possible options are:

KindStr	Description
Pedestrian	motion model of pedestrian
Course	single-point motion (direction is defined by course angle)
Ball	ball motion model considering road surface
4Wheel	single-track model with roll and pitch model (see section 21.4.4 '4Wheel')
2Wheel	single-track model with specific roll model (see section 21.4.5 '2Wheel')

Example `Traffic.0.Motion.Kind = 4Wheel`

Traffic.<Id>.Motion.mass = ***value***

Specifies the overall object mass [kg]. Default: 1300. Not required for the pedestrian model.

21.4.1 Pedestrian

The pedestrian is always oriented directly in the direction which is defined by the course angle (vector addition of longitudinal velocity vector and lateral velocity vector) and always moves forwards. A negative longitudinal velocity results in movement against the path. The object orientation is always vertical in z direction of inertial frame Fr0. The pedestrian model does not support the maneuver "Autonomous driving".

21.4.2 Course

The traffic object is always oriented directly in direction of the course angle (vector addition of longitudinal velocity vector and lateral velocity vector). The model uses various filters to reduce possible jumps in the road coordinates, e.g. smoothing direction vectors, usage of PID controller to smooth the lateral road coordinate tRoad and the usage of a PT1 filter to smooth the Euler angle.

21.4.3 Ball

The ball motion model is used to simulate a rolling or a hopping ball on a road. The model considers the road surface including all road bumps for the contact point calculation. Interaction with other objects is not supported. The ball model supports a start condition for the model activation, but does not have special maneuvers. The model is disabled if the ball leaves the road.

Traffic.<Id>.Motion.z_off = ***value***

Specifies the initial z-offset of the ball above road level [m]. Default: 0.

21.4.4 4Wheel

The motion kind 4Wheel uses a linear single-track model with 2 DOF: the yaw angle φ and the sideslip angle β. Furthermore, a dynamic roll model and pitch model with 1 DOF exists. Both are used to calculate the roll and pitch motion by the integration of the lateral and longitudinal acceleration. The pitch model is only activated for vehicle which are not longer than 13 meters. The Implicit Euler method is used as integration method for all models.

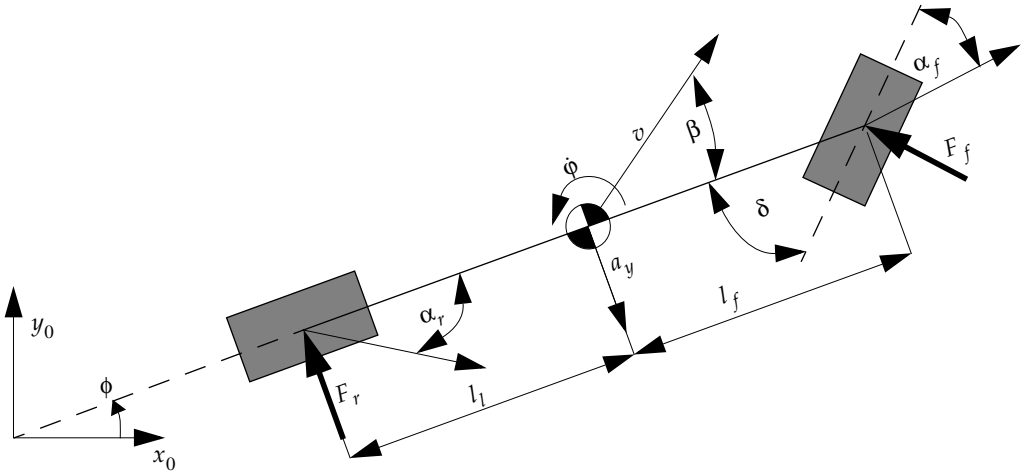


Figure 21.7: single-track model

Relevant variables are:

α_f	front slip angle	C_f	front cornering stiffness rate
α_r	rear slip angle	C_r	rear cornering stiffness rate
a_y	lateral acceleration	F_f	front side force
β	sideslip angle	F_r	rear side force
δ	steer angle	l_f	distance between CoM and front axle
ϕ	yaw angle	l_r	distance between CoM and rear axle
$\dot{\varphi}$	yaw velocity	J_Z	moment of inertia about z axis
		m	vehicle mass
		v	velocity of CoM

The equations of motion of the single track model:

$$\dot{\beta} = -\frac{C_f + C_r}{m \cdot v} \cdot \beta + \frac{m \cdot v^2 - C_r \cdot l_r - C_f \cdot l_f}{m \cdot v^2} \cdot \dot{\phi} - \frac{C_f}{m \cdot v} \cdot \delta \qquad \text{(EQ 467)}$$

$$\ddot{\phi} = -\frac{C_r \cdot l_r - C_f \cdot l_f}{J_Z} \cdot \beta - \frac{C_r \cdot l_r^2 - C_f \cdot l_f^2}{J_Z \cdot v} \cdot \dot{\phi} + \frac{C_f \cdot l_f}{J_Z} \cdot \delta \qquad \text{(EQ 468)}$$

The traffic object is always moving in the direction of the course angle with the velocity defined in the longitudinal maneuver.

The steer angle is defined as the input to the single-track model. The lateral deviation between the marker position and the path is regulated by a steer angle pilot controller and a PID controller. The marker position is defined at the front axis for driving forward and

behind the rear axis for driving backwards by default. In addition, a steer angle as function of the course orientation exists at low speeds. The maximum value of the steer angle corresponds to the maximum wheel angle and the minimum turning circle.

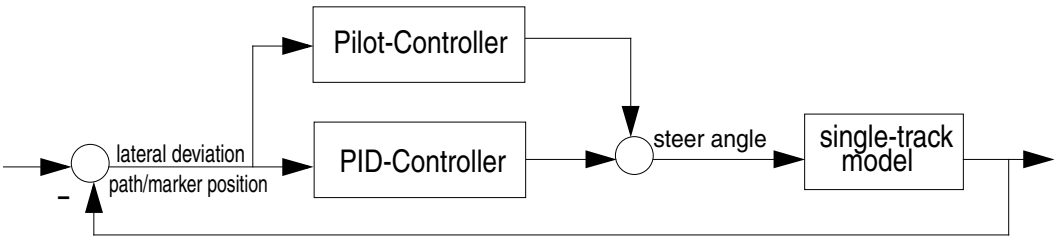


Figure 21.9: steer angle pilot controller and PID-controller

Traffic.<Id>.Motion.Overhang = ***value_front value_rear***

Specifies front/rear overhang [m]. Default: length/6.

Traffic.<Id>.Motion.I = ***Ixx Iyy Izz***

Specifies the moment of inertia about the x, y, z-axis [kgm^2].
Default: Ixx = Izz/4, Iyy = Izz, Izz = 2600.

Traffic.<Id>.Motion.Cf = ***value***
Traffic.<Id>.Motion.Cr = ***value***

Specifies the front and rear cornering stiffness rate [N/rad].
Default: Cf = 7e4, Cr = 16e4.

Traffic.<Id>.Motion.C_roll = ***value***

Optional. Specifies the roll stiffness rate [Nm/rad].
Default: mass*(h/2+zOff)/(0.6°).

Traffic.<Id>.Motion.D_roll = ***value***

Optional. Specifies the roll damping rate [Nms/rad]. Default: C_roll/10.

Traffic.<Id>.Motion.C_pitch = ***value***

Optional. Specifies the pitch stiffness rate [Nm/rad]. Default: mass*(h/2+zOff)/(0.2°).

Traffic.<Id>.Motion.D_pitch = *value*

Optional. Specifies the pitch damping rate [Nms/rad]. Default: C_pitch/10.

Traffic.<Id>.Motion.SteerCtrl.Ang_max = *value*

Optional. Specifies the max. steer angle [deg]. Default: 40.

Traffic.<Id>.Motion.SteerCtrl.p = *P-value*
Traffic.<Id>.Motion.SteerCtrl.i = *I-value*
Traffic.<Id>.Motion.SteerCtrl.d = *D-value*

Optional. Specifies the parameter of the steer angle PID controller.
Default: P= 5, I= 2, D= 0.5.

Traffic.<Id>.Motion.lenFr12Mar_forw = *value*
Traffic.<Id>.Motion.lenFr12Mar_back = *value*

Specifies the marker position for driving forwards[m].
Default: l - OverhangFront.

Optional. Specifies the marker position for driving backwards [m].
Default: OverhangRear - 0.2.

21.4.5 2Wheel

The 2Wheel motion kind uses the same single-track model with steer angle regulation as the 4Wheel motion kind. However, this motion kind consists of a specific roll model which includes the correction of the z coordinate of the CoM because of object leaning into curves depending on the lateral acceleration, but does not include a pitch model.

All the parameters are similar to the parameters of the 4Wheel motion kind, except for the default values of the following.

Traffic.<Id>.Motion.I = *0 0 Izz*

Specifies the moment of inertia about the z-axis [kgm^2]. Default: 100.

21.5 Maneuver

The traffic object can perform complex driving maneuvers based on a sequence of minimaneuvers - in analogy to the ego vehicle. Each minimaneuver description is divided into a longitudinal and a 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 a superimposed maneuver in longitudinal and lateral direction.

21.5.1 Maneuver Reference

RoadEval XYZ

RoadEval XYZ describes a motion along the defined route. The longitudinal speed is the speed of the CoM (Center of Mass) and is aligned in the direction of the course. The traffic object moves in the direction of the route.

RoadEval ST

RoadEval ST enables a motion which is independent from the defined route, e.g parking vehicles or pedestrians crossing the road. The coordinates sRoad and tRoad have to be specified to define the path of the object and to calculate the longitudinal and lateral velocity.

Vs here is basically the internally calculated derivative of sRoad.

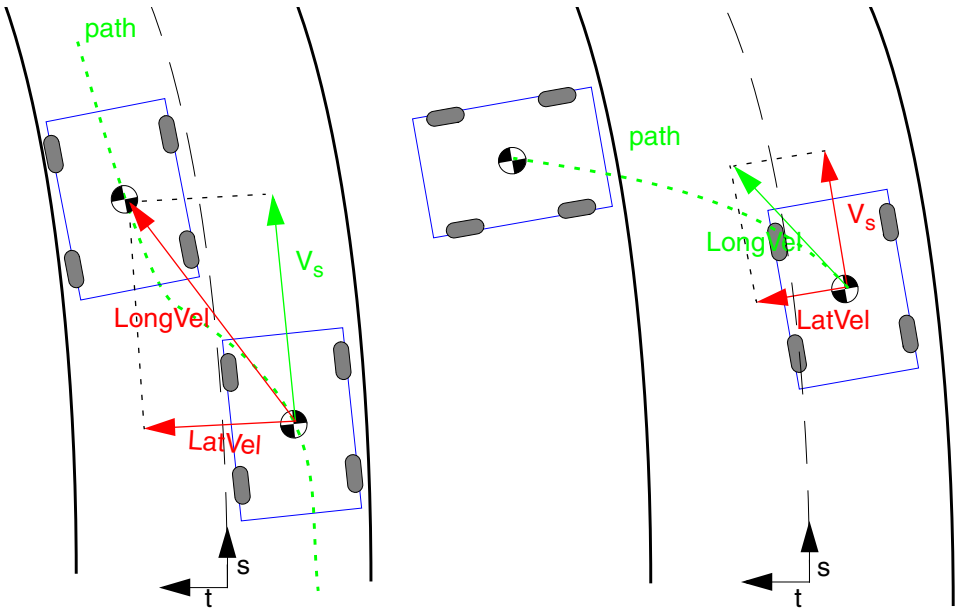


Figure 21.11: Velocity vectors of the two motion references
(left: RoadEval XYZ, right: RoadEval ST; red: input, green: calculated)

21.5.2 General Maneuver Parameters

The motion type for the Traffic maneuver is given by the standard prefix "FreeMotion" or "Man" along with the default key Traffic.<Id> depending upon the selected motion mode.

Traffic.<Id>.UpdRate = *value*

If *Traffic.UnderSampling = 1* in the SimParameters file, the traffic position and motion calculation is updated with a defined rate between 50 and 1/DeltaT [Hz], otherwise if *Traffic.UnderSampling = 0* or *Free Motion = 1* the update rate 1/DeltaT is used. Default: 200. If the update rate parameter is set to -1, the value is calculated depending on object velocity within the range 50...200 Hz.

The update rate is also set at the value 99999 for the whole maneuver if one of the following maneuver steps is used:

- maneuver steps using *Input From File* (Velocity, Position, Lateral Position)
- maneuver steps for user defined input (Acceleration, Velocity, Position, Lateral Position)
- *Path Points*

Traffic.<Id>.FreeMotion = *bool*

Parameter for the activation of Free Motion instead of motion by defined maneuver. Default: 0 (= Maneuver).

Traffic.<Id>.Man.N

Number of maneuver steps. Default: 0.

Traffic.<Id>.Man.StartCondition = *string*

User defined start condition using Realtime Expressions for the maneuver.

Traffic.<Id>.Man.TreatAtEnd = *TreatAtEnd*

Define the behavior of the traffic object at the end of the maneuver

Treat at End	Description
Hide	traffic object disappears
FreezePos	traffic object remains at the position of the final minimaneuver
FreezeVel	traffic object keeps moving with the same, constant velocity of the last minimaneuver

Example `Traffic.0.Man.TreatAtEnd = Hide`

A second prefix is built up for each minimaneuver, it is the traffic prefix followed by the string “Man” and the minimaneuver number. The keys of all parameters for each maneuver start with this prefix.

For the parameters of the minimanuevers the following prefix is built:

<ManPre> := Traffic.<Id>.Man.<ManNo>

<ManPre>.Limit = *parameter value*

Each minimaneuver is characterized by a duration of the maneuver step. Below, the options available to define the duration of a maneuver step are listed:

Duration	Parameter	Description
Time	t	Duration of the maneuver step [s]. Value<0 or empty means that the duration is endless.
Abs. Time	t_abs	Maneuver is aborted when the absolute simulation time reaches the defined time [s].
Distance	s	Distance of the maneuver step [m]. Value<0 or empty means that the duration is endless.
Position	s_abs	Maneuver lasts until the object reaches the defined absolute road distance [m].

Example `Traffic.2.Man.1.Limit = t 4`

<ManPre>.Visib = *bool*

This parameter gives the user the ability to make a traffic object appear and disappear. For a boolean value of 1 the traffic object is visible. When the object is invisible, it is not detected by the sensor. Default: 1.

<ManPre>.EndCondition = *string*

User defined end condition using Realtime Expressions for manuever step.

21.5.3 Longitudinal Motion

The following options to specify the longitudinal motion of the traffic object are available:

Longitudinal Motion	Description
Distance	The distance specified should be covered at the end of the maneuver step.
Velocity	Target velocity reached at the end of this maneuver step with a constant acceleration.
Acceleration	The acceleration specified is constantly applied.
Position	The absolute road position should be reached at the end of the maneuver step.
User Velocity	Longitudinal velocity described by a user function in C-code or by DVA
User Accel.	Longitudinal acceleration described by a user function in C-code or by DVA
User Position	The absolute road position described by a user function in C-code or by DVA
IFF Velocity	Longitudinal velocity described by Input From File (see section 21.8.1 'Longitudinal Maneuver')
IFF Position	The absolute road position described by Input From File (see section 21.8.1 'Longitudinal Maneuver')
Path Points	Movement along the defined <S=sRoad, T=tRoad> points, independent of the route.
Autonomous	Autonomous driving (see section 21.6 'Autonomous Driving')

Example `Traffic.1.Man.1.LongDyn = v 72`

<ManPre>.LongDyn = *s* *value*

Selection of the longitudinal maneuver *Distance* and specifies the distance [m].

<ManPre>.LongDyn = *v* *value*

Selection of the longitudinal maneuver *Velocity* and specifies the target velocity which follows the unit as specified in "Traffic.SpeedUnit". The maneuver ends when the target velocity is reached.

<ManPre>.LongDyn = *a* *value*

Selection of the longitudinal maneuver *Acceleration* and specifies the longitudinal acceleration [m/s^2].

<ManPre>.LongDyn = *s_abs value*

Selection of the longitudinal maneuver *Position* and specifies the absolute road position [m].

<ManPre>.LongDyn = *vUser*

Selection of the longitudinal maneuver *User Velocity*.

The velocity can be manipulated via modification of correspond DVA quantities or directly in the C-Code using the parameters:

```
Traffic.<Name>.LongVel
```

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

<ManPre>.LongDyn = *aUser*

Selection of the longitudinal maneuver *User Acceleration*.

The acceleration can be manipulated via modification of correspond DVA quantities or directly in the C-Code using the parameters:

```
Traffic.<Name>.LongAcc
```

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

<ManPre>.LongDyn = *s_abs_User*

Selection of the longitudinal maneuver *User Position*.

The position can be manipulated via modification of correspond DVA quantities or directly in the C-Code using the parameters:

```
Traffic.<Name>.sRoad
```

The target parameter must be set in each simulation cycle, since the signal ist input and output.

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

<ManPre>.LongDyn = *auto value*

Selection of the longitudinal maneuver *Autonomous Driving* and specifies the target cruising velocity in the unit as specified in "Traffic.SpeedUnit".

<ManPre>.LongDyn = *s_abs_PathST* *bool <Absolut>* *T_RampUp*
<ManPre>.PathST: sRoad tRoad rz drz/ds dTime Vel

Selection of the longitudinal maneuver *PathST*.

The ST points (sRoad, tRoad) correspond to the center of the rear axis when using the 4Wheel and 2Wheel motion kind and correspond to the CoM when using the course or pedestrian motion kind.

<Absolut>: When 1, ST points are considered as absolute instead of relative to current motion (0). When 0, the current velocity and orientation are used which means that the first velocity and orientation values defined in the data table are ignored.

Optional. If <Absolut> = 1, reaching the first point of velocity and orientation of the data table using a pre-polynomial in the time specified in <T_RampUp>.

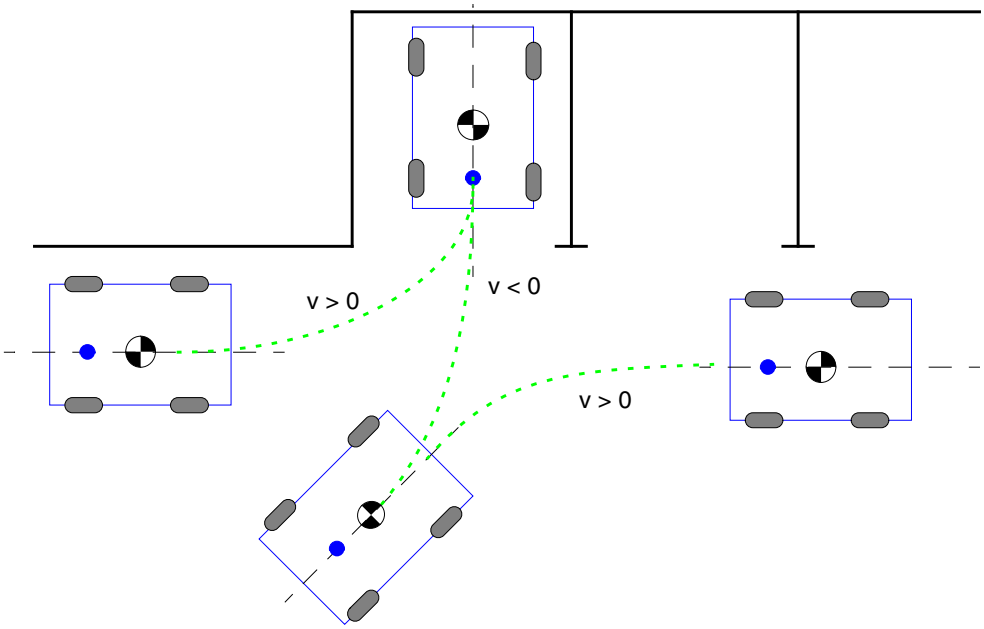


Figure 21.13: Parking maneuver as an example for absolute ST points without usage of a pre-polynomial

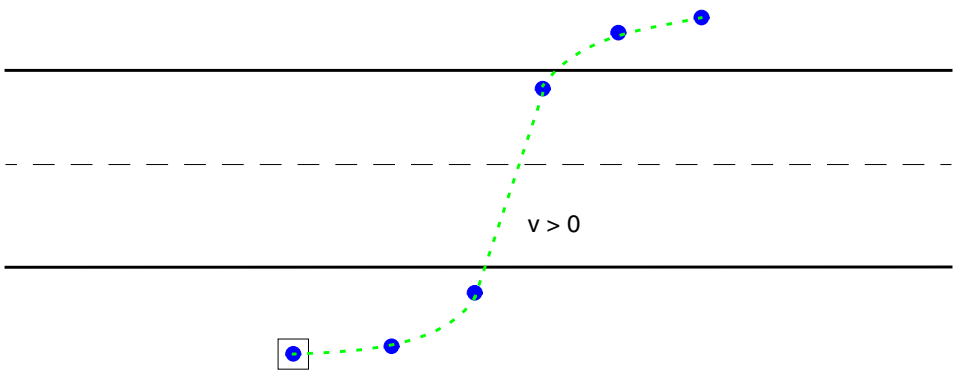


Figure 21.14: Pedestrian crosses the street as an example for relative ST points

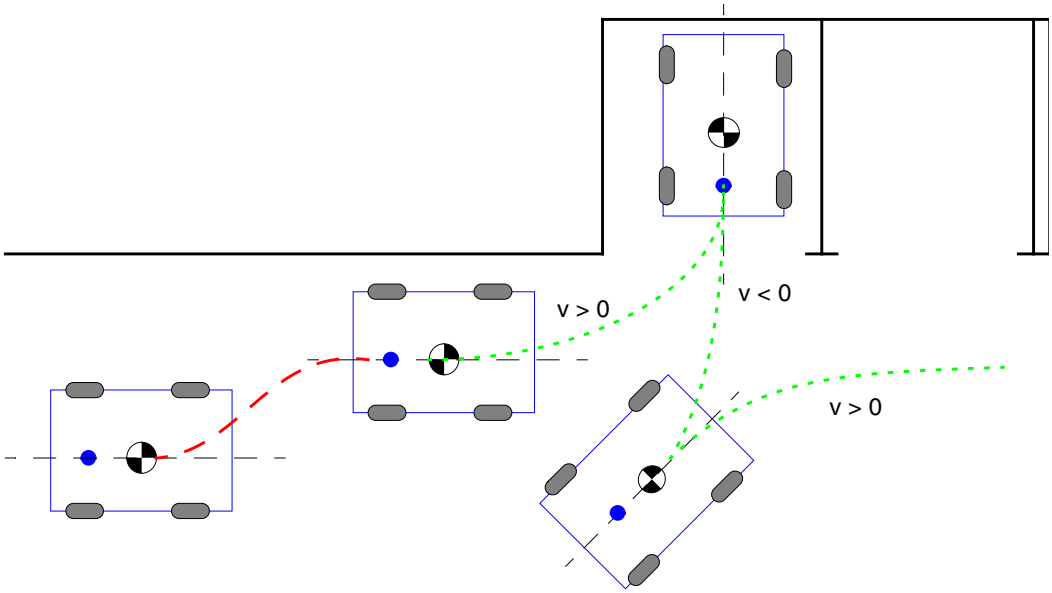


Figure 21.15: Parking maneuver with the usage of a pre-polynomial [red]

The ST - data table consists of six columns.The first and second column are the sRoad and tRoad coordinates [m]. The third column rz is the orientation based on the direction of the path at point ST [deg] followed by the orientation change [deg/m]. The next column is the delta time [s] and has to start with zero. The last column is the velocity at point ST [km/h].

Syntax

Infofile table mapping with 6 columns
<sRoad[m]> <tRoad[m]> <rz[deg]> <drz/ds[deg/m]> dTime[s] <Vel[km/h]>

Example

```
90   -4   0   0   0   5
105   6   90  0   20  0
105   6   90  0   5   0
85   -6   0   0   25  0
...
```

Two points can be the same, but not the first two and the velocity has to be set to zero at both points. This describes a standstill for the defined delta time.

If this is the first maneuver, the option of absolute coordinates is selected and the ramp time is set to zero, then the first ST-point is used as the start position of the object.

A delta time value can be set to -1, what estimates internally the time using a mean velocity and the distance between the two points. Additionally the total minimaneuver time is logged.

<ManPre>.LongDyn = ***vIFF*** ***tOffset*** ***bool <Restart>*** ***bool <Smooth>***

Selection of the longitudinal maneuver *IFF Velocity* (see section 21.8.1 'Longitudinal Maneuver').

The maneuver ends if the end of the time channel is reached. If this is the first maneuver, the first velocity value is used as start velocity of the traffic object. More than one object can use the same velocity channel.

<tOffset>: time channel offset [s]
<Restart>: When 1, it means relative maneuver time instead of absolute time.
<Smooth>: When 1, it means smoothing with PT2 filter.

<ManPre>.LongDyn = *s_abs_IFF tOffsetbool <Restart> bool <Smooth>*

Selection of the longitudinal maneuver *IFF Position* (see section 21.8.1 'Longitudinal Maneuver').

The maneuver ends if the end of the time channel is reached. If this is the first maneuver, the first value is set as start position of the traffic object. More than one object can use the same position channel.

<tOffset>: time channel offset [s]
<Restart>: When 1, it means relative maneuver time instead of absolute time.
<Smooth>: When 1, it means smoothing with PT2 filter.

<ManPre>.LongSup = *sinus Amp T*

Apart from the nominal longitudinal motion an additional sine motion with amplitude <Amp> [m] and time period <T> [s] can be superimposed.

21.5.4 Lateral Motion

The following options to specify the lateral motion of the traffic object are available:

Lateral Motion	Description
Lat. Offset	Offset in lateral direction, measured from the initial road position of the traffic object.
Lat. Position	Absolute deviation from to route or path, reached at the end of the maneuver step.
User Lat. Position	The absolute lateral road position described by a user function in C-code or by DVA
IFF Lat. Position	The absolute lateral road position described by Input From File (see section 21.8.2 'Lateral Maneuver')
Lane Change	Change of driving lane
Path Change	Change of driving path
Follow Path	Follow path with a tolerated lateral deviation

Example `Traffic.1.Man.3.LatDyn = y_abs 8.0`

<ManPre>.LatDyn = *y value*

Selection of the lateral maneuver *Lateral Offset* and specifies the relative deviation [m].

<ManPre>.LatDyn = *y_abs value*

Selection of the lateral maneuver *Lateral Position* and specifies the absolute deviation [m].

<ManPre>.LatDyn = *y_abs_User*

Selection of the lateral maneuver *User Lat. Position*.

The transverse position can be manipulated via modification of corresponding DVA quantity *Traffic.<Name>.tRoad* or directly in the C-Code *tTrafficObj *Obj->tRoad*.

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

<ManPre>.LatDyn = *y_abs_IFF <tOffset> bool <Restart> bool <Smooth>*

Selection of the lateral maneuver *IFF Lat. Position* (see section 21.8.2 'Lateral Maneuver').

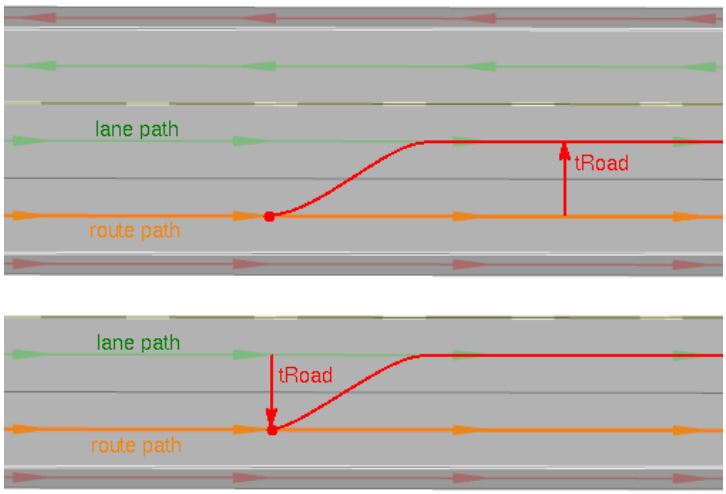
The maneuver ends if the end of the time channel is reached. If this is the first maneuver, the first value is used as start transverse position of the traffic object. More than one object can use the same transverse position channel.

<tOffset>: time channel offset [s]
<Restart>: When 1, it means relative maneuver time instead of absolute time.
<Smooth>: When 1, it means smoothing with PT2 filter.

<ManPre>.LatDyn = *LaneChange 0 <deltaLane> bool<OnlyDrvLane> <mode>* or <ManPre>.LatDyn = *LaneChange 1 <absoluteLane> <mode>*

Selection of the lateral maneuver *Lane Change*. Specifies the target driving lane as a absolute lane, e.g. <absoluteLane>="L0" or "R1" or relative to the current driving lane by <deltaLane>. If <deltaLane> is greater than zero, a change to left otherwise a change to right is done. When <OnlyDrvLane> is set to 1, only driving lanes (usual lane for vehicle) are used. Crossing a footpath or cordoned area is not allowed. The lane change is only done if the width of the target lane is at least 0.2 meters wider than the object witdh.

The function supports two modes: keep the current path and change the lane by offset (<mode>="offset", default) or leave the current route and change to the new lane path via dynamic path (<mode>="new"). Change for the second mode works only for lane in the same direction.



mode **offset**:
keep the current path (e.g. route path) and change the lane by constant offset

mode **new**:
leave the current path to new lane path to follow the new lane using dynamic path

Figure 21.16: Two modes for lane change maneuver

If the target driving lane corresponds to the current driving lane (<deltaLane> is zero or absolute lane equals to current lane), the minimaneuver corrects the lateral position to the current lane middle.

<ManPre>.LatDyn = *PathChange 0 <Route: Id or Name> bool<UsePath>* or
<ManPre>.LatDyn = *PathChange 1 <List of path object Ids: ObjId0 ... ObjId4>*

Selection of the lateral maneuver *Path Change* with two modes like for the starting condition: change to a route path (mode=0, default) or to a dynamic path (mode=1).

The route path is specified by route internal Id or route name and a flag if the route driving path or route reference line is used.
The dynamic path is specified by a list of one or up to five paths (lane, user or connector path) specified by their global road obejct Id=*ObjIdx.*

If the time duration is zero, then a change is done only to a new path. If another duration setting is selected or the time value is greater than zero, then a change can be moved to a new path.

<ManPre>.LatDyn = *FollowPath <TolDev>*

Selection of the lateral maneuver *Follow Path*: follows the current route or dynamic path.

In the case of driving on a dynamic path, the maneuver identifies randomly the subsequent path(s) depending on weight factor and valid vehicle class of the outgoing paths.

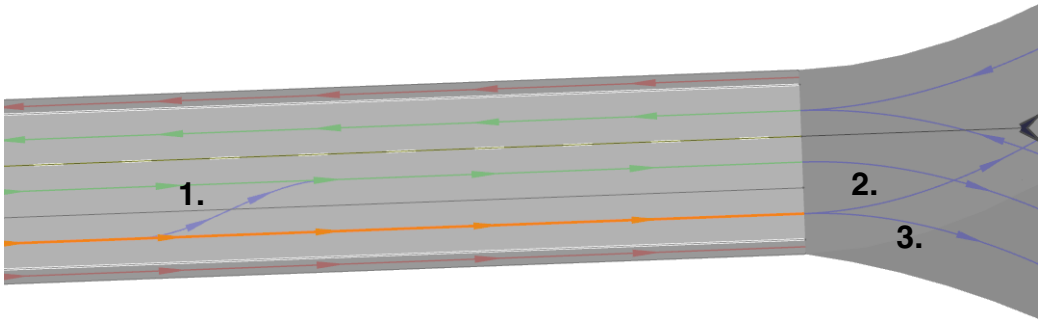


Figure 21.17: Selection of the subsequent path

For both cases, driving on a route or dynamic path, the traffic object can change randomly the lateral position referred to the path within the specified tolerated deviation *<TolDev>*. The value=-1 means to hold the lateral position from last maneuver step constantly and value=0 means to follow exactly the path with a zero offset. Default: 0.5m.

<ManPre>.LatSup = *sinus Amp T*

Apart from the nominal lateral motion an additional sine motion with amplitude <Amp> [m] and time period <T> [s] can be superimposed.

21.6 Autonomous Driving

The following parameters 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 speed limit signs, traffic lights and other traffic objects but also the ego-vehicle.

Three predefined settings are available for autonomous driving (see Predefined Driver Settings).

21.6.1 Cruising

Traffic.<Id>.AutoDrv.AxMax = ***value***

Specifies the max. long. acceleration [m/s^2].

Traffic.<Id>.AutoDrv.AxMin_CC = ***value***

Specifies the max. long. deceleration for cruising functionality [m/s^2].

Traffic.<Id>.AutoDrv.AyMax = ***value***

Specifies the max. lateral acceleration [m/s^2]. Default: 4 (2 if mass > 4000 kg).

Traffic.<Id>.AutoDrv.p = ***value***

Specifies the P-Ratio of the control unit [-]. Default: 0.41.

Parameters to specify the regulation of the target velocity before a curve as function of the chosen lateral acceleration.

Traffic.<Id>.AutoDrv.CPV.p = ***value***

Optional. Specifies the parameter of the P-controller for the target velocity. Default: 6.0.

Traffic.<Id>.AutoDrv.CPV.sBuiltUp = ***value***

Optional. Specifies the filter constant for the velocity as function of the distance [m]. Default: 3.0.

Traffic.<Id>.AutoDrv.CPV.AxMin = *value*

Optional. Specifies the max. deceleration before a curve [m/s^2]. Default: -3.0.

21.6.2 Following

If the traffic object is equipped with a following function (works like an ACC sensor), you can also define the sensitivity parameters $TauD$, $TauV1$, $TauV2$ and the reference relative velocity.

$TauD$ 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. 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 $vRef12$, the factor $TauV2$ shall be small to get a more dynamic control behavior.

In this model the target acceleration Ax is calculated following the book "Handbuch Fahrerassistenzsysteme, Winner, Hakuli, Wolf, 2009, chapter 32".

$$\Delta vRel = \frac{\Delta Dist - SafeStopDist - DesrTGap \cdot LongVel}{TauD} + \Delta Vel \qquad \text{(EQ 469)}$$

If the relative velocity $\Delta vRel$ is less than or equal the reference relative velocity $vRef12$ then

$$Ax = \frac{\Delta vRel}{TauV1} \qquad \text{(EQ 470)}$$

else

$$Ax = \left(\frac{vRef12}{TauV1} + \frac{|\Delta vRel| - vRef12}{TauV2}\right) \cdot \mathrm{sgn}(\Delta vRel) \qquad \text{(EQ 471)}$$

Three predefined settings are available for autonomous driving (see Predefined Driver Settings).

Traffic.<Id>.AutoDrv.AxMin_ACC = *value*

Specifies the max. long. deceleration for following functionality [m/s^2]. Default: -10.

Traffic.<Id>.AutoDrv.SafeStopDist = *value*

Specifies the stop safety distance [m].

Traffic.<Id>.AutoDrv.PreviewDist = ***value***

Specifies the maximum preview distance [m]. Default: 200.

Traffic.<Id>.AutoDrv.DesrTGap = ***value***

Specifies the target time gap to traffic object in front [s].

Traffic.<Id>.AutoDrv.TauD = ***value***

Specifies the relative distance $TauD$ [s].

Traffic.<Id>.AutoDrv.TauV1 = ***value***
Traffic.<Id>.AutoDrv.TauV2 = ***value***

Specifies the relative velocity $TauV1$ and $TauV2$ [s].

Traffic.<Id>.AutoDrv.vRef12 = ***value***

Specifies the reference relative velocity [m/s]. Default: 1.

21.6.3 Lane Change

The maneuver *Autonomous* detects if the path, along which the object moves (white object in the picture below), crosses a new regular lane and other pathes on the new lane (e.g. motorway access). In the case of a detection, the object searches for a suitable gap, accelerates or stops for the next gap. Merging of lanes (with no preference for each lane) is also considered by this mode.

Functional limitation:
Both objects (itself and target) must drive along the path without/small lat. deviation.

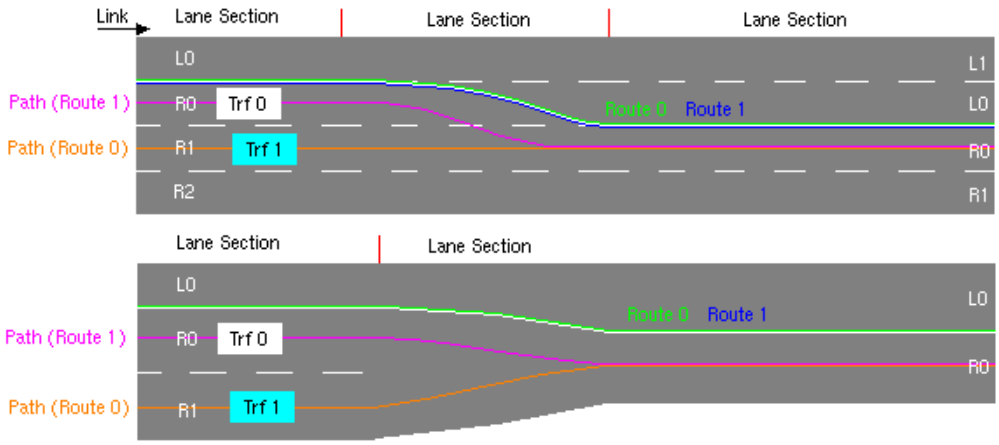


Figure 21.18: Lane change along path or merging lanes

Traffic.<Id>.AutoDrv.Cautious =*value*

Specifies the cautiousness of the autonomous driver for the lane change functionality[0...1]. Default: 0.5.

Predefined Driver Settings

Following table illustrates the default parameters of the predefined driver settings:

Driver set	AxMax	AxMinCC	AyMax	p-Ratio	StopDist	TGap	TauD	TauV1	TauV2	Cautious
Driver set for normal passenger car vehicle class										
defensive	1.0	-1.0	2.0	0.25	6.0	2.0	6.0	4.0	2.0	0.75
normal	2.0	-2.0	4.0	0.41	3.0	1.5	4.0	2.6	1.3	0.5
aggressive	4.0	-3.0	5.0	0.55	1.5	1.0	2.0	1.5	0.8	0.25
Driver set for large vehicle class (van, caravan, truck, semi truck, bus, coach)										
defensive	0.8	-0.8	1.5	0.2	8.0	2.5	8.0	5.0	3.0	0.6
normal	1.5	-1.5	3.0	0.35	4.5	1.75	5.0	3.2	1.7	0.4
aggressive	2.5	-2.5	4.0	0.45	2.0	1.25	3.0	2.0	1.0	0.2

21.6.4 Longitudinal Vehicle Dynamics

To grant a more realistic acceleration of the traffic object, the vehicle parameters are taken into account.

Traffic.<Id>.LVD.AxMax = *value*

Specifies the max. acceleration [m/s^2].

Traffic.<Id>.LVD.vMax = *value*

Specifies the max. velocity [km/h].

Traffic.<Id>.LVD.Pmax = *value*

Specifies the max. driving power [kW].

Traffic.<Id>.LVD.tBuildUp = *value*

Specifies the response time of the object acceleration [s].

21.7 Free Motion

The traffic object's position can be manipulated either via modification of the corresponding Direct Variable Access (DVA) quantities, directly using the parameters of the C-code interface or using Input From File.

Direct Variable Access

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
```

C - Code

The correspond variables are in the header file *Traffic*.h in the struct tTrafficObj:

```
tTrafficObj *TrfObj = Traffic_GetByTrfId(<TrafficId>);
//Optional with global object Id:
//TrfObj = Traffic_GetByObjId(<ObjId>);
TrfObj->t_0[0]
TrfObj->t_0[1]
TrfObj->t_0[2]
TrfObj->r_zyx[0]
TrfObj->r_zyx[1]
TrfObj->r_zyx[2]
```

Input from File

The positioning information can be stored in an external file and can be used as an input for free motion, for details see section 21.8.3 'Free Motion'.

21.8 Input From File

Traffic.IFF.FName = ***FName***

File name with relative path to directory of the file. Only one file is read in for all traffic objects.

Traffic.IFF.Time.Name = name

Specifies the name of the time channel. Only one time channel is required for all traffic objects. The time channel has to start with zero and the values have to increase monotonously.

21.8.1 Longitudinal Maneuver

Longitudinal Velocity from Input From File

Traffic.<Id>.IFF.Vel.Name = ***name***
Traffic.<Id>.IFF.Vel.Factor = ***value***
Traffic.<Id>.IFF.Vel.Offset = ***value***

Specifies the name, the amplification factor and offset of the velocity channel [m/s] for each traffic object. Default: Factor = 1.0, Offset = 0.0.

Absolute Road Position from Input From File

Traffic.<Id>.IFF.sRoad.Name = ***name***
Traffic.<Id>.IFF.sRoad.Factor = ***value***
Traffic.<Id>.IFF.sRoad.Offset = ***value***

Specifies the name, the amplification factor and offset of the absolute road coordinate sRoad channel [m] for each traffic object. Default: Factor = 1.0, Offset = 0.0.

Traffic.<Id>.IFF.YawAng.Name = ***name***
Traffic.<Id>.IFF.YawAng.Factor = ***value***
Traffic.<Id>.IFF.YawAng.Offset = ***value***

Optional. Specifies the name, the amplification factor and offset of the yaw angle channel [rad] for each traffic object. Default: Factor = 1.0, Offset = 0.0.

21.8.2 Lateral Maneuver

Absolute Lateral Road Position from Input From File

Traffic.<Id>.IFF.tRoad.Name = ***name***
Traffic.<Id>.IFF.tRoad.Factor = ***value***
Traffic.<Id>.IFF.tRoad.Offset = ***value***

Specifies the name, the amplification factor and offset of the absolute lateral road position tRoad coordinate channel [m] for each traffic object. Default: Factor = 1.0, Offset = 0.0.

21.8.3 Free Motion

Parameters for the Free Motion channel <FM_x> (FM_x:= FM_tx, FM_ty, FM_tz, FM_rx, FM_ry, FM_rz):

Traffic.<Id>.IFF.<FM_x>.Name =	***name***
Traffic.<Id>.IFF.<FM_x>.Factor =	***value***
Traffic.<Id>.IFF.<FM_x>.Offset =	***value***

Specifies the name, the amplification factor and offset of the Free Motion channel <FM_x>. The units are [m] for FM_tx, FM_ty, FM_tz and [rad] for FM_rx, FM_ry, FM_rz.

The value is only accepted if the time channel is not ended. It only works with the absolute time. More than one object can use the same Free Motion channels.
Default: Factor = 1.0, Offset = 0.0.

21.9 User Accessible Quantities for Traffic

Please refer to section 24.16 'Traffic'.

Chapter 22

Geographic Coordinate System (GCS)

22.1 Introduction

CarMaker has the possibility to describe any position in the global road frame *Fr0* as a location point on the earth's surface. This could be interesting in GPS or navigation applications. To describe a position on the earth's surface the most common coordinate system *Geographic Coordinate System (GCS)* is used. This system consists of lines of geographic (geodetic) latitude φ, longitude λ and elevation h. Lines of equal latitude (parallels) and lines of equal longitude (meridians) form the graticule when projected onto a map plane.

Figure 22.1shows a location point P on the earth's surface in the geographic coordinate system:

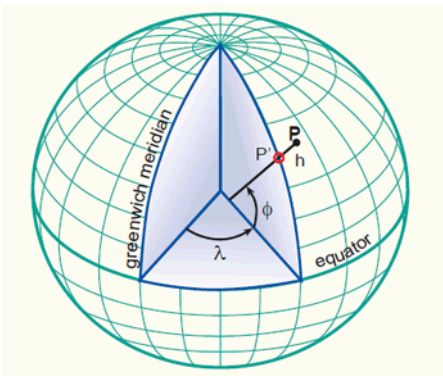


Figure 22.1: Geographic coordinate system

The latitude φ of a point P is the angle between the ellipsoidal normal through P' and the equatorial plane. Latitude is zero on the equator and increases towards the two poles to maximum values of $\phi = 90°$ at North Pole and $\phi = -90°$ at South Pole.

The longitude λ is the angle between the meridian ellipse which passes through Greenwich and the meridian ellipse containing the point in question. It is measured in the equatorial plane from the meridian of Greenwich ($\lambda = 0°$) either eastwards through $\lambda = 180°$ or westwards through $\lambda = -180°$.

The elevation (height) h is the vertical distance of P above the ellipsoid. It is measured in distance units along the ellipsoidal normal from the point to the ellipsoid surface.

Angular unit of geographical coordinates

There are several formats for the angular units of geographic coordinates. CarMaker uses the format *Decimal Degrees (DD)*, for example $48.56789°$, $-12.34981°$.

Reference earth ellipsoid (geodetic datum)

CarMaker uses as the reference ellipsoid *WGS-84 (World Geodetic System 1984)*. This geodetic datum is the most common datum for GPS applications and has the following parameters:

- Equatorial radius a: 6378137m
- Polar radius b: 6356752.3142m
- Inverse flattening f: 298.257223563

22.2 Projection onto the road plane

There are several methods how to project the 3D ellipsoid graticule onto a 2D plane. CarMaker currently supports two methods: the fast computed method *FlatEarth* and the complex method *GaussKrueger*. The GCS projection method is set in the TestRun infofile with:

Road.GCS.Projection = ***Method***

Optional. Specifies the projection method of geographical coordinates longitude, latitude to the 2D road plane. Possible modes are *GaussKrueger* and *FlatEarth*. Default: *FlatEarth*. This parameter is set automatically if a digitalized, from GCS coordinates converted and imported road is used.

22.2.1 GCS reference point

To locate the road frame *Fr0* origin on the earth's surface, a GCS reference point is required. This can be any point on the road or outside the road, whose cartesian coordinates in the virtual road frame (x_R, y_R, z_R) and the corresponding geographic coordinates on the earth's surface (ϕ_R, λ_R, h_R) are known.

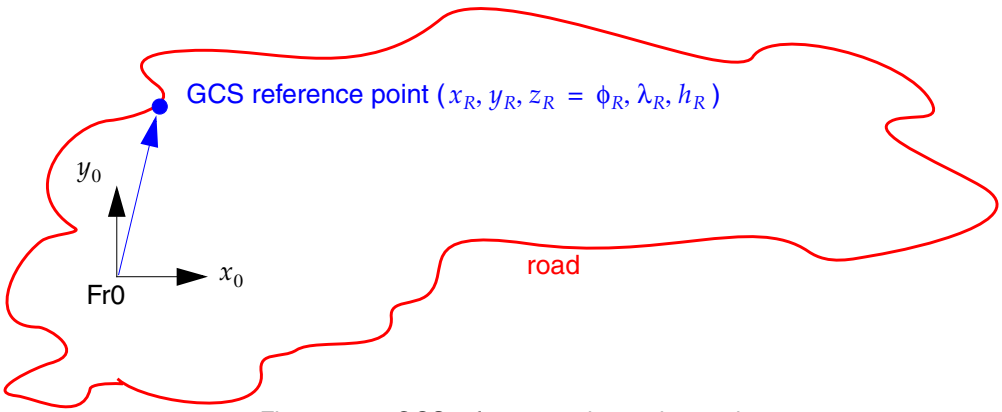


Figure 22.2: GCS reference point on the road

To minimize the projection error, it is recommended, if possible, to place the GCS reference point somewhere in the middle of the road circumference.

The GCS reference point is parameterized in the TestRun infofile.

Road.GCS.RefPos_0 = tx ty tz

Specifies the global position of the reference point in the road frame *Fr0*.
Unit: [m]. Default: 0 0 0.

Road.GCS.RefPos_GCS = Lat Long Elev

Specifies the geographic coordinates (latitude, longitude, elevation) of the reference point. Unit: [deg deg m]. Default: 0 0 0. The GCS module converts automatically if this key is set.

22.2.2 Theory of projection method *FlatEarth*

The projection method *FlatEarth* neglects the earth curvature and assumes the earth to be flat around the GCS reference point. This fast computed method is sufficient for roads with small extension.

Figure 22.3 illustrates the calculation of the latitude of any point P referring to the GCS reference point.

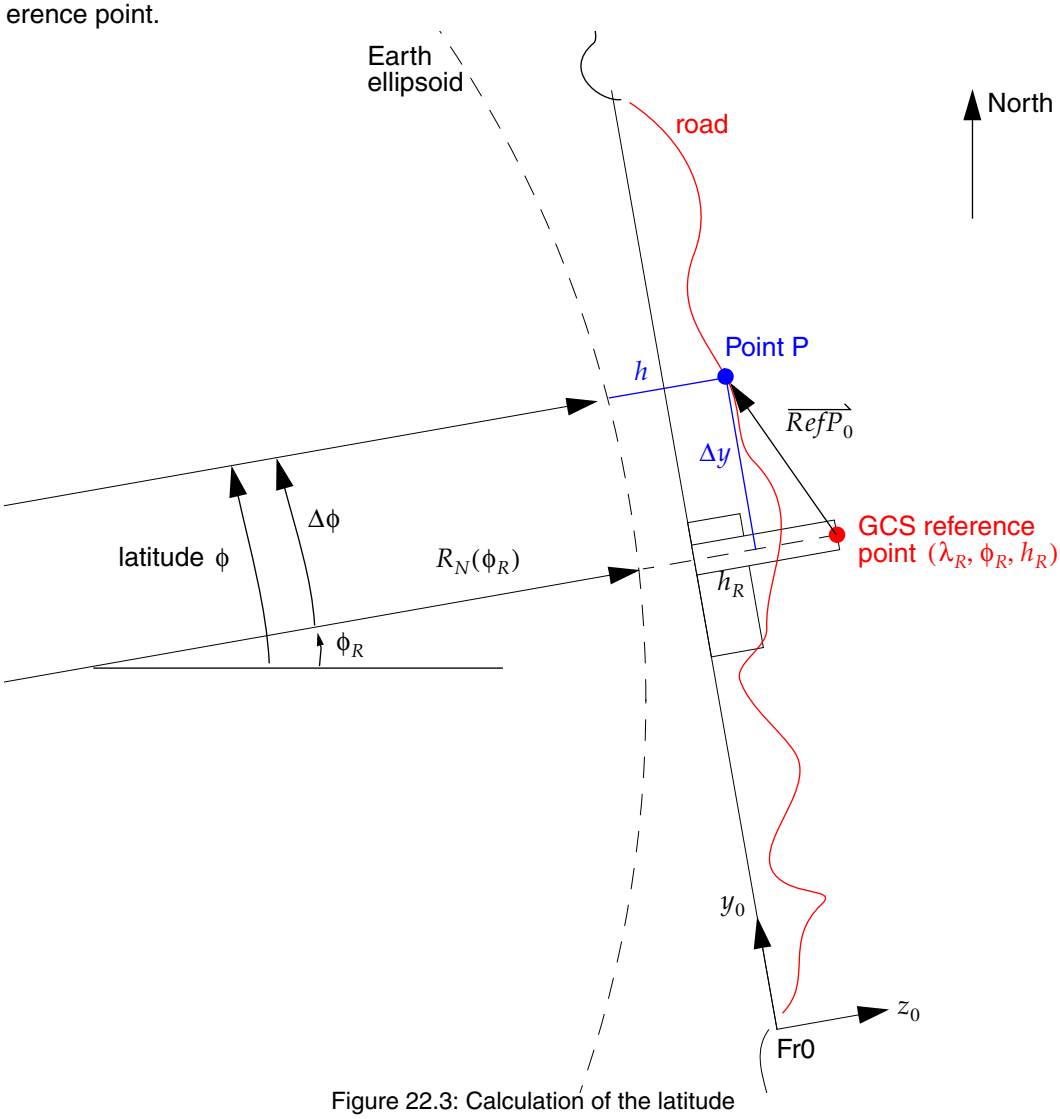


Figure 22.3: Calculation of the latitude

First the relative position $\overrightarrow{RefP_0}$ of the point P to the GCS reference point in the road frame is determined. The elevation h of the point P is calculated as follows:

$$h = h_R + \overrightarrow{RefP}[z]_0 \quad \text{(EQ 472)}$$

The relative latitude $\Delta\phi$ can be determined as follows:

$$\Delta\phi = \operatorname{asin}\left(\frac{\Delta y}{R_N(\phi_R) + h}\right) \approx \frac{\Delta y}{R_N(\phi_R) + h} = \frac{\overrightarrow{RefP}[y]_0}{R_N(\phi_R) + h} \quad \text{(EQ 473)}$$

$R_N(\phi_R)$ is the radius of the earth ellipsoid in north direction and depends on the latitude in the GCS reference point.

The latitude ϕ of the point P becomes:

$$\phi = \phi_R + \Delta\phi \quad \text{(EQ 474)}$$

In the same manner the longitude in the point P can be calculated as follows:

$$\lambda = \lambda_R + \frac{(\overrightarrow{RefP}[x]_0)}{(R_E(\phi_R) + h) \cdot \cos(\phi_R)} \quad \text{(EQ 475)}$$

$R_E(\phi_R)$ is the radius of the earth ellipsoid in east direction and depends also on the latitude in the GCS reference point. The factor $\cos(\phi_R)$ considers the smaller radius with increasing latitude.

22.2.3 Theory of projection method *GaussKrueger*

The CarMaker projection method *GaussKrueger* is based on the Gauß-Krüger-Projection, which is known as a transverse Mercator map projection. This is a conformal isogonal projection, which maintains angular relationships.

The idea of this projection method is to split the earth's surface along the longitude into 120 zones, each with $3°$ width, see Figure 22.4. The zones are projected onto a plane map maintaining the original distension along the equator (east-axle) and along each zone meridian (north-axle: $0°$, $3°$, $6°$, ...). The Gauß-Krüger system is similar to the *Universal Transverse Mercator (UTM)* system, but the second uses zones with $6°$ width.

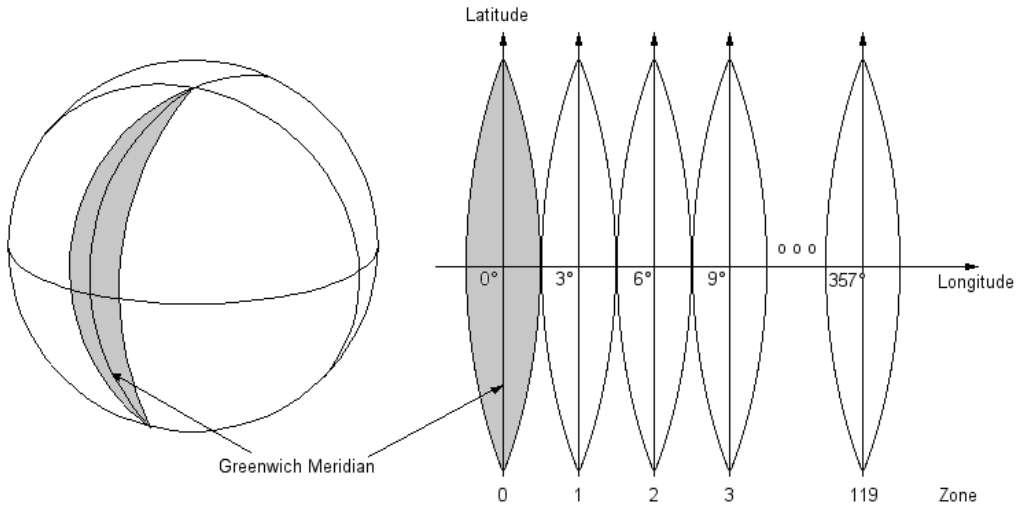


Figure 22.4: Gauß-Krüger-Projection

Figure 22.5 demonstrates the Gauß-Krüger coordinate system. The value along the north-axle is called *Northing*. It begins at the equator with zero. The value along the east-axle is called *Easting*. First the easting of the point P relative to the central meridian of the zone N ist calculated. To avoid negative easting values, the central meridian begins with 500km. Finally to identify in which zone the point is locating, the expression N*1000km is added.

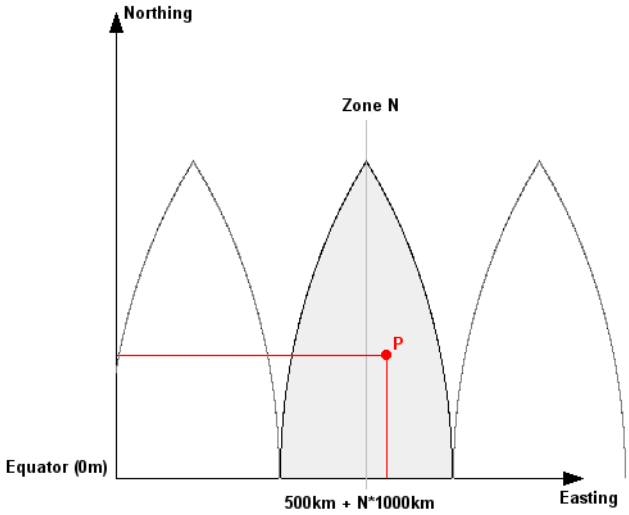


Figure 22.5: Gauß-Krüger-Coordinates

Treatment in CarMaker

First of all, during initialization, the Gauß-Krüger coordinates (northing, easting) and the corresponding zone of the GCS reference point are determined, see Figure 22.6. The determined zone is now used for all road points even if the point could contact next the zone:

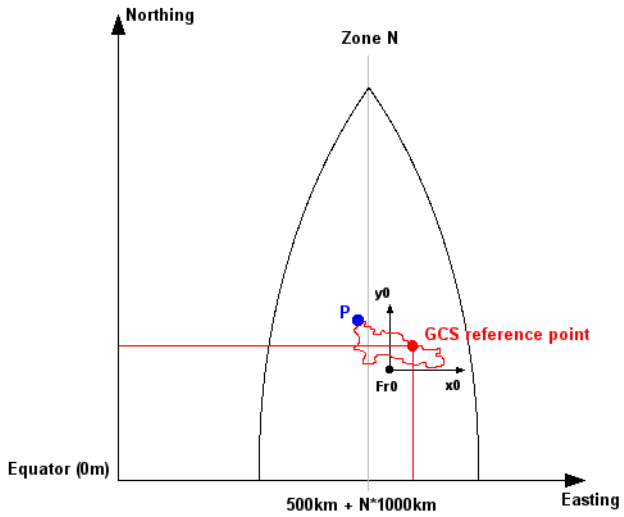


Figure 22.6: Treatment in CarMaker using Gauß-Krüger-Projection

During the simulation the relative position of point P referring to the reference point in virtual road frame *Fr0* is known. With this information the Gauß-Krüger coordinates of the point P can be determined:

$$\begin{bmatrix} N \\ E \\ h \end{bmatrix}_{GK} = \begin{bmatrix} N_R \\ E_R \\ h_R \end{bmatrix}_{GK} + \begin{bmatrix} \Delta y \\ \Delta x \\ \Delta z \end{bmatrix}_0 \qquad \text{(EQ 476)}$$

In the end with the Gauß-Krüger algorithm the GCS coordinates of the point P are calculated. The details of the algorithm aren't shown:

$$\begin{bmatrix} \phi \\ \lambda \\ h \end{bmatrix}_{GCS} = f_{GK \rightarrow GCS}\left(\begin{bmatrix} N \\ E \\ h \end{bmatrix}_{GK} \right) \quad \text{(EQ 477)}$$

22.3 User Accessible Quantities

Following global positions in road frame *Fr0* are converted in geographic coordinates:

- *Car.Road.t[x,y,z]* -> *Car.Road.GCS.[Long, Lat, Elev]*
- *Traffic.<Name>.t[x,y,z]* -> *Traffic.<Name>.GCS.[Long, Lat, Elev]*

The traffic quantities are converted if the GCS coordinates calculation of traffic objects is set in the *Data/Config/SimParameters* file, described in section 3.1.16 'Geographic coordinate system'.

Chapter 23

Special Modes

This section explains exceptional mode features.

23.1 Standby Mode

With the Standby Mode the vehicle position and orientation can be frozen at any instant of time, on a plane or inclined road surface.

The main reason for introducing such a mode is that it provides a way to synchronize the vehicle models to the powertrain testbed.

The Standby Mode can be activated using *DirectVariableAccess* with the quantity *Car.StandbyMode.* Following states of the mode are available:

State	Description
0	Off: Standby Mode is disabled
1	Pause: the global vehicle velocity and position in frame Fr0 keep constant; the rotation velocity along x- and y-axis become zero; the rotation angles keep constant
2	Synchronization: identical to State=1, but the velocity in x-direction of vehicle is calculated. Thus synchronization with a testbed is possible

23.2 SlotCar Mode

The Slotcar Mode consists in keeping constant (with respect to local road-coordinate) the lateral position of the vehicle. Switching the SlotCar Mode active will prevent the vehicle from lane departure by pure kinematics.

The SlotCar Mode intervenes and corrects the lateral position if the lateral position of the SlotCar reference point is outside of the tolerance range, see Figure 23.1.

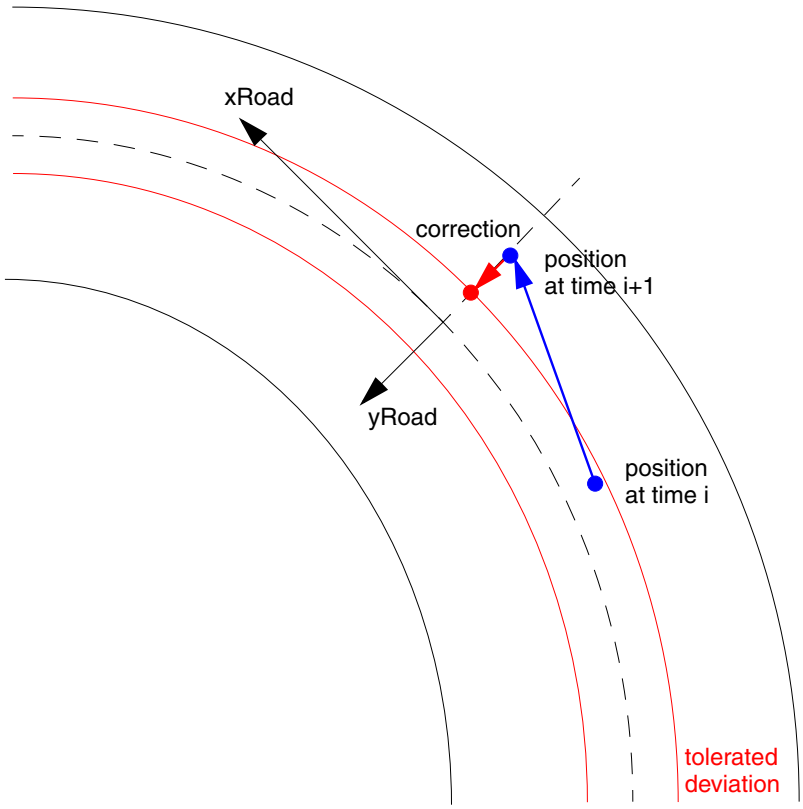


Figure 23.1: Correction with SlotCar Mode outside of the tolerated deviation range

Following parameter in the Vehicle File describes the position of the SlotCar reference point:

SlotCar.Pos_x = *Value*

For the SlotCar reference point only the x-Position is required, y-Position and z-Position are zero. This parameter specifies the x-Position in vehicle fixed frame Fr1. Unit: [m].
Default: x-Position of the *front axle middle point*.

The tolerated deviation along the road centerline can be specified in two ways: using *DirectVariableAccess* with the quantity *Car.SlotCar.Deviation* or using the following parameter in SimParameters File:

SlotCar.Deviation = *Value*

Specifies the initial tolerated deviation along the road centerline within the SlotCar Mode correction is disabled. Unit: [m]. Default: -1.

If the value of deviation is negative, the SlotCar Mode is disabled.

The tolerated deviation should be high enough if the driver with a high corner cutting coefficient should work properly without important interventions of the SlotCar Mode. The same should be considered for driving on left or right driving lane.

23.3 Pylon Detection

The Pylon Detection module observes during simulation if the vehicle passes through a pylon alley and detects if a pylon of a pylon alley marker is hit. This functionality is activated automatically, if the road contains at least one pylon alley marker.

The following assumptions are made for the calculation:

- detection works only in a two-dimensional x-y-plane (the global z-position of the pylons must be zero)
- the two pylons of the marker are considered as two one-dimensional points
- for the calculation of the jut-out the vehicle outer skin is taken (see Figure 23.2).

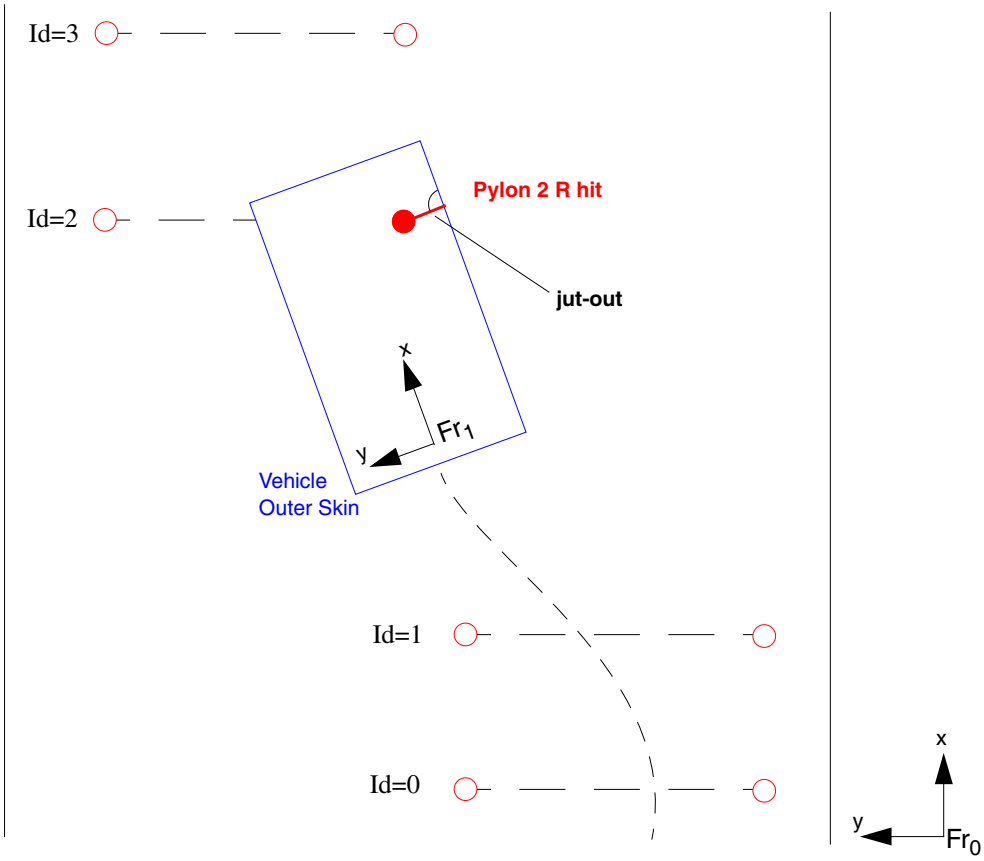


Figure 23.2: Pylon detection

Everytime a pylon is hit, a Scratchpad note *PylonHit* will be created. It contains the following information:

Note-Variable	Description
id	id of the pylon alley marker (beginning with 0)
side	0: if left pylon is hit; 1: if right pylon is hit
sRoad	road distance of the pylon alley marker
pos_0.x	global x-position of the hit pylon
pos_0.y	global y-position of the hit pylon

This Scratchpad note can be handled by ScriptControl or the TestManager. All Scratchpad notes are also saved in the TestRun result file (.erg.info file). An upper limit of Scratchpad notes sent during a single TestRun can be defined in the *SimParameters-File (section 3.1.13 'Pylon detection')*.

The current maximum jut-out of all hit pylons can be read from quantity *PylonDetect.CurJutOut*. The quantity *PylonDetect.MaxJutOut* describes the maximum jut-out over the whole TestRun.

Chapter 24

User Accessible Quantities

For a better understanding of the naming conventions of UAQ´s please refer to section 1.4 ’CarMaker Naming Conventions’. If there is no entry in the column "Name C-Code" for a UAQ, this means the c-code variable is not accessible for the user (private variable).

24.1 General

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
DeltaT	DeltaT	s	Duration of completed calculation cycle.
DeltaTPeak		s	Maximum duration of completed calculation cycle
Implicit		-	Sample number in IPGControl
Time	SimCore.Time	s	Simulation-time of the current TestRun Time starts after initialization process is finished. This time may be accelerated or delayed during non-realtime simulations.
Time.Global	TimeGlobal	s	Simulation-time since the start of the application. This time may be accelerated or delayed during non-realtime simulations.
Time.WC	SimCore.TimeWC	s	“Wall-clock-time” This is the real time elapsed since the start of the application. In non-realtime simulations this time may differ from Time.Global.

24.1.1 TCPU

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
TCPU.AposEvalSend	SimCore.TCPU.AposEvalSend	s	Time-consumption for function AposEvalSend during the current calculation cycle (float)
TCPU.AposPoll	SimCore.TCPU.AposPoll	s	Time-consumption for function AposPoll during the current calculation cycle (float)
TCPU.Brake	SimCore.TCPU.Brake	s	Time-consumption for calculation of the Brake module during the current calculation cycle (float)
TCPU.Sensors	SimCore.TCPU.Sensors		Time-consumption for calculation of the sensor modules during the current calculation cycle (float)
TCPU.DrivMan	SimCore.TCPU.DrivMan	s	Time-consumption for calculation of the DrivMan module during the current calculation cycle (float)
TCPU.In	SimCore.TCPU.In	s	Time-consumption for calculation of the IO_In module during the current calculation cycle (float)
TCPU.Out	SimCore.TCPU.Out	s	Time-consumption for calculation of the IO_Out module during the current calculation cycle (float)
TCPU.PowerTrain	SimCore.TCPU.PowerTrain	s	Time-consumption for calculation of the PowerTrain module during the current calculation cycle (float)
TCPU.Total	SimCore.TCPU.Total	s	Total time-consumption for *calculation* of the current cycle. This should not be confused with DeltaT (float)
TCPU.Traffic	SimCore.TCPU.Traffic	s	Time-consumption for calculation of the Traffic module during the current calculation cycle (float)
TCPU.Trailer	SimCore.TCPU.Trailer	s	Time-consumption for calculation of the Trailer module during the current calculation cycle (float)
TCPU.User	SimCore.TCPU.User	s	Time-consumption for calculation of the User module during the current calculation cycle (float)
TCPU.Vehicle	SimCore.TCPU.Vehicle	s	Time-consumption for calculation of the Vehicle module during the current calculation cycle (float)
TCPU.VehicleControl	SimCore.TCPU.VehicleControl	s	Time-consumption for calculation of the VehicleControl module during the current calculation cycle (float)

24.1.2 TGPU

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
TGPU.GPUSensor.<No>.Total		s	Time-consumption for GPUSensor calculation cycle (float)

24.1.3 Misc

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
Log.nError	Log_nError	-	Counter for log file errors (long)
Log.nLog	Log_nLog	-	Counter for log file messages (long)
Log.nWarn	Log_nWarn	-	Counter for log file warnings (long)
SC.MemFree		-	For internal use only (unsigned integer)
SC.MemUsed		-	For internal use only (unsigned integer)
SC.MemUsed.TRNo		-	For internal use only (unsigned integer)
SC.Start.No	SimCore.Start.No	-	Number of TestRun started and stopped since the start of the application (unsigned integer)
SC.State	SimCore.State	-	Simulation state (integer)
SC.TAccel	SimCore.TAccel	-	Time acceleration factor for offline simulation

24.2 Environment

Name UAQ	Name C-Code	Name CM4SL	Unit	Info
Env.AirDensity	Env.AirDensity	Env AirDensity	kg/m^3	Environment air density
Env.AirHumidity	Env.AirHumidity	Env AirHumidity	-	Environment air relative humidity
Env.AirPressure	Env.AirPressure	Env AirPressure	bar	Environment air pressure
Env.SolarRadiation	Env.SolarRadiation	Env SolarRadiation	W/m^2	Environment solar radiation
Env.Temperature	Env.Temperature	Env Temperature	K	Environment air temperature
Env.RainRate	Env.RainRate	-	mm/h	Environment rain rate
Env.VisRangeInFog	Env.VisRangeInFog	-	m	Environment visual range in fog
Env.Time	Env.Time	Env Time	hours	Time of day (0..24)
Env.YearDay	Env.yDay	Env yDay	days	Day of the year
-	-	Env PoI_GCS Elev Env PoI_GCS Long Env PoI_GCS Lat	m rad rad	Global position for PoI, expressed in GCS coordinates (geodetic elevation, longitude, latitude)
-	-	Env sRoad	m	Vehicle route / path coordinate
Env.WindVel_ext.x Env.WindVel_ext.y Env.WindVel_ext.z	Env.WindVel_ext[0] Env.WindVel_ext[1] Env.WindVel_ext[2]	-	m/s	Additional user defined wind in inertial frame
Env.WindVel_tot.x Env.WindVel_tot.y Env.WindVel_tot.z	Env.WindVel_tot[0] Env.WindVel_tot[1] Env.WindVel_tot[2]	Env WindVel_tot x Env WindVel_tot y Env WindVel_tot z	m/s	Total environment global wind in inertial frame (global + road wind marker + external)

24.3 Driving Maneuvers

Name UAQ	Name C-Code	Name CM4SL	Unit	Info (data type, if not double)
DM.ax.Trgt	-	-	m/s^2	Target acceleration used for Speed Profile
DM.Brake	DrivMan.Brake	DrivMan Brake	-	Brake/decelerator activity, relative pedal force (0..1)
DM.BrakePark DM.Handbrake	DrivMan.BrakePark	DrivMan Brake Park	-	Park brake activity (0..1)
DM.Clutch	DrivMan.Clutch	DrivMan Clutch	-	Clutch activity (0..1)
DM.Gas	DrivMan.Gas	DrivMan Gas	-	Gas/throttle/accelerator activity (0..1)
DM.GearNo	DrivMan.GearNo	DrivMan GearNo	-	Gear number (integer)
DM.GearNo.Trgt	DrivMan.GearNoTrgt	-	-	Target gear number set at the beginning of the shifting procedure during manual shifting (integer)
DM.Key	DrivMan.Key	DrivMan Key	-	Vehicle key position (integer): 0: Key out 1: Key in, power off 2: Key in, power accessory 3: Key in, power on 4: Key in, starter active
DM.LaneOffset	DrivMan.LaneOffset	-	m	Lateral offset from center line of vehicle's current lane for IPGDriver, that is provided by the user
DM.Lap.No		-	-	Lap number (racing mode) (integer)
DM.Lap.Time		-	s	Lap time (racing mode)
DM.Lights.FogFront	DrivMan.Lights.FogFront	-	-	Front fog light on (boolean)
DM.Lights.FogRear	DrivMan.Lights.FogRear	-	-	Rear fog light on (boolean)
DM.Lights.Hazard	DrivMan.Lights.Hazard	-	-	Hazard warning light on (boolean)
DM.Lights.HighBeam	DrivMan.Lights.HighBeam	-	-	High beam: 0=Off;1=On; 2=Automatic
DM.Lights.Indicator	DrivMan.Lights.Indicator	-	-	Turn indicator: -1= Right; 0=Off; 1=Left
DM.Lights.MainLight	DrivMan.Lights.MainLight	-	-	Main light switch: 0=Off; 1=Parking light; 2=Low beam; 3=Automatic
DM.ManDist	DrivMan.ActualMan.Dist	-	m	Mini maneuver distance
DM.ManNo	DrivMan.ActualMan.No	-	-	Mini maneuver number (integer)
DM.ManTime	DrivMan.ActualMan.Time	-	s	Mini maneuver time
DM.OperationState_trg	DrivMan.OperationState_trg	-	-	Target operation state for the vehicle operator (integer): 0: Absent (leave the vehicle) 1: Power off 2: Power accessory 3: Power on 4: Driving
DM.OperatorActive	DrivMan.OperatorActive	-	-	Vehicle operator active (boolean)
DM.SelectorCtrl	DrivMan.SelectorCtrl	DrivMan SelectorCtrl	-	Automatic gear selector position (integer): -9 = position P (corresponds to gear number -9) -1 = position R (corresponds to gear number -1...-8) 0 = position N (corresponds to gear number =0) 1 = position D (corresponds to gear number >0) 2 = position M (manual gear selection / Manumatic)
DM.Shifting	DrivMan.Shifting	-	-	Gear shift in progress (boolean)
DM.SpeedLimit	DrivMan.SpeedLimit	-	m/s	Current speed limit set by road marker
DM.SpeedOffset	DrivMan.SpeedOffset	-	m/s	Longitudinal speed offset for IPGDriver (considered in the maneuvers 'IPGDriver' and 'Speed Profile')
DM.SST	DrivMan.SST	DrivMan SST	-	Powertrain start-stop button (boolean): 0=off, 1=on
DM.Steer.Ang	DrivMan.Steering.Ang	DrivMan Steering Ang	rad	Steering angle at steering wheel
DM.Steer.AngAcc	DrivMan.Steering.AngAcc	DrivMan Steering AngAcc	rad/s^2	Steering angle acceleration at steering wheel
DM.Steer.AngVel	DrivMan.Steering.AngVel	DrivMan Steering AngVel	rad/s	Steering angle velocity at steering wheel
DM.SteerBy	DrivMan.ActualMan.SteerBy	-	-	Steering mode (integer): 1=by angle, 2=by torque
DM.Steer.SinusFreq		-	Hz	Steering angle frequency for sine maneuver
DM.Steer.Trq	DrivMan.Steering.Trq	DrivMan Steering Trq	Nm	Steering torque at steering wheel

Name UAQ	Name C-Code	Name CM4SL	Unit	Info (data type, if not double)
DM.StopDist	DrivMan.StopDist	-	m	Distance to the next stop sign -1=no stop sign ahead
DM.StopTime	DrivMan.StopTime	-	s	Remaining time standing at current stop sign -1=no stop sign ahead
DM.TriggerPoint.Dist		-	m	Trigger point distance
DM.TriggerPoint.Id		-	-	Trigger point Id (integer)
DM.TriggerPoint.Time		-	s	Time measured at trigger point
DM.UserSignal_<i>	DrivMan.UserSignal[]	DrivMan UserSignal	-	User defined signal <i> from vehicle operator to powertrain control (e.g. seat belt, vehicle door, seat sensor)
DM.vdelta.Trgt		-	m/s	Difference between current velocity and target velocity used for Speed Control, Speed Profile and Input From File
DM.v.Trgt		-	m/s	Target velocity used for Speed Control, Speed Profile and Input From File (value can be manipulated via DVA only for Speed Control)
DM.v.Trgt_LimitLo		-	m/s	Lower target velocity limit (used for Speed Profile)
DM.v.Trgt_LimitHi		-	m/s	Upper target velocity limit (used for Speed Profile)

24.4 Driver

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<kind> := FOL (following), ONC (oncoming), OTR (overtaking right)			
Driver.Adapt.Phase		-	Current learning phase during the basic learning procedure (integer)
Driver.Brake		-	Brake/decelerator activity, relative pedal force (0..1)
Driver.Brake_UDrvIn	UserDriver.In.Brake	-	Brake/decelerator activity from user driver (float, 0..1)
Driver.BrakePark		-	Park brake activity (0..1)
Driver.BrakePark_UDrvIn	UserDriver.In.BrakePark	-	Park brake activity from user driver (float, 0..1
Driver.Clutch		-	Clutch activity (0..1)
Driver.Clutch_UDrvIn	UserDriver.In.Clutch	-	Clutch activity from user driver (float, 0..1)
Driver.Gas		-	Gas/throttle/accelerator activity (0..1)
Driver.Gas_UDrvIn	UserDriver.In.Gas	-	Gas/throttle/accelerator activity from user driver (float, 0..1)
Driver.GearNo		-	Gear number set by driver (integer)
Driver.GearNo_UDrvIn	UserDriver.In.GearNo	-	Gear number set by user driver (integer)
Driver.GearNoTrgt		-	Target gear number set at the beginning of the shifting procedure (integer)
Driver.Lat.dy		m	Lateral deviation from the desired static course
Driver.Lat.passive		-	Passive flag of lateral controller (boolean)
Driver.Long.dv		m/s	Speed deviation from the desired static velocity
Driver.Long.passive		-	Passive flag of longitudinal controller (boolean)
Driver.ReCon.Accel		m/s^2	Max. admitted acceleration for speeding up the vehicle while driving backwards.
Driver.ReCon.ADAS_LD_Coef		-	ADAS Interface: -1=max. brake, 0=neutral, +1=max. throttle
Driver.ReCon.ADAS_LD_T		s	ADAS Interface: time delay of pedal actuation
Driver.ReCon.Decel		m/s^2	Max. admitted deceleration e.g. for stopping the vehicle
Driver.ReCon.DriveChar.CF_axy			Correction factor for axmin, axmax, aymax
Driver.ReCon.DriveChar.CF_GGExp			Correction factor for GG-Exponent
Driver.ReCon.DriveChar.iChange			Change flag; no continuos change possible (boolean)
Driver.ReCon.DriveMode		-	Driving mode (unsigned integer) 1=forwards, 2=stop, 3=backwards, ...
Driver.ReCon.JunDir		-	Change direction at next junction: +1=to left, -1=to right
Driver.ReCon.Speed		km/h	Target speed for the selected driving mode
Driver.ReCon.StopDist		m	Distance for stopping (e.g. at traffic sign) -1=no stop requested
Driver.ReCon.StopTask		-	Driver stop task (1..5) see ipgdriver.h

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
Driver.ReCon.StopTime		s	Time to stop (e.g. at a traffic sign) -1=no stop requested
Driver.ReCon.Task		-	Driver's current task for longitudinal dynamics (0..32) see ipgdriver.h
Driver.ReCon.Task_lat		-	Driver's current task for longitudinal dynamics (-6..13)
Driver.ReCon.Trf_Consider		-	Flag: Consider traffic (boolean)
Driver.ReCon.Trf_<kind>.dDist		m	Distance to target traffic object
Driver.ReCon.Trf_<kind>.dSpeed		m/s	Relative speed of target traffic object
Driver.ReCon.Trf_<kind>.ObjId		-	Identification number of target object Id: - value (integer) =-1 stands for no detection - value individual to each scenario - value not contiguous, so it is unsuitable for use as an array index
Driver.ReCon.Trf_<kind>.Targ_Dtct		-	Flag: Target traffic object is detected by driver (boolean)
Driver.ReCon.Trf_Overtake		-	Flag: Overtake traffic objects (boolean)
Driver.ReCon.Trf_Overtake_LaneOffset		m	Offset to left free lane for traffic overtaking
Driver.SelectorCtrl		-	Automatic gear selector position (integer): -9 = position P (corresponds to gear number -9) -1 = position R (corresponds to gear number -1...-8) 0 = position N (corresponds to gear number =0) 1 = position D (corresponds to gear number >0) 2 = position M (manual gear selection / Manumatic)
Driver.SetGearNo		-	Change gear number to the desired gear (integer)
Driver.Shifting		-	Driver is shifting (boolean)
Driver.Steer.Ang		rad	Steering angle at steering wheel
Driver.Steer.AngAcc		rad/s^2	Steering angle acceleration at steering wheel
Driver.Steer.AngVel		rad/s	Steering angle velocity at steering wheel
Driver.Steer.Trq		Nm	Steering torque at steering wheel
Driver.UDrv.Active	UserDriver.In.Active	-	Flag: User driver is activated (boolean)
Driver.WaitFlag			Flag: Wait to start (boolean)

24.5 Vehicle

Name UAQ	Name C-Code	Name CM4SL	Frame	Unit	Info (data type, if not double)
<pos> := FL, FR, RL, RR <pre> := Vehicle.<pos>					
-	Vehicle.Trq_DL2Bdy1[0] Vehicle.Trq_DL2Bdy1[1] Vehicle.Trq_DL2Bdy1[2]	Vhcl Trq_DL2Bdy1 x Vhcl Trq_DL2Bdy1 y Vhcl Trq_DL2Bdy1 z		Nm	Drive torque supported by vehicle body Fr1A
-	Vehicle.Trq_DL2Bdy1B[0] Vehicle.Trq_DL2Bdy1B[1] Vehicle.Trq_DL2Bdy1B[2]	Vhcl Trq_DL2Bdy1B x Vhcl Trq_DL2Bdy1B y Vhcl Trq_DL2Bdy1B z		Nm	Drive torque supported by vehicle body Fr1B
-	Vehicle.Trq_DL2BdyEng[0] Vehicle.Trq_DL2BdyEng[1]	Vhcl DL2BdyEng x Vhcl DL2BdyEng y		Nm	Drive torque supported by vehicle engine body
-	Vehicle.AeroMarkerPos[0] Vehicle.AeroMarkerPos[1] Vehicle.AeroMarkerPos[2]	Vhcl AeroMarkerPos x Vhcl AeroMarkerPos y Vhcl AeroMarkerPos z	Fr0	m	Aero marker position in global frame (Fr0)
Vhcl.Distance	Vehicle.Distance	Vhcl Distance		m	Distance the vehicle (center of mass) has travelled since the start of simulation
Vhcl.Engine.rotv	Vehicle.Engine_rotv	-		rad/s	Engine rotation speed
Vhcl.Fr1B.rx Vhcl.Fr1B.ry	Vehicle.Fr1B_r_xy[0] Vehicle.Fr1B_r_xy[2]	-	Fr1	rad	Rotation angles of flexible vehicle body with rotation order XY
Vhcl.Fr1.x Vhcl.Fr1.y Vhcl.Fr1.z	Vehicle.Fr1A.t_0[0] Vehicle.Fr1A.t_0[1] Vehicle.Fr1A.t_0[2]	Vhcl Fr1_Pos x Vhcl Fr1_Pos y Vhcl Fr1_Pos z	Fr0	m	Position of vehicle frame origin (Fr1) in global frame (Fr0)
Vhcl.GearNo	Vehicle.GearNo	-		-	Current gear number (integer)
Vhcl.Hitch.x Vhcl.Hitch.y Vhcl.Hitch.z	Vehicle.Hitch.t_0[0] Vehicle.Hitch.t_0[1] Vehicle.Hitch.t_0[2]	Vhcl Hitch t_0 x Vhcl Hitch t_0 y Vhcl Hitch t_0 z	Fr0	m	Hitch position in global frame (Fr0)
Vhcl.Ignition	Vehicle.Ignition	Vhcl Ignition		-	Vehicle ignition (boolean)
Vhcl.OperationError	Vehicle.OperationError	Vhcl OperationError		-	Current vehicle operation error from powertrain control (integer)
Vhcl.OperationState	Vehicle.OperationState	Vhcl OperationState			Current vehicle operation state from powertrain contro (integer): 0: Absent (leave the vehicle) 1: Power off 2: Power accessory 3: Power on 4: Driving
Vhcl.Pitch	Vehicle.Pitch	Vhcl Pitch		rad	Vehicle pitch angle
Vhcl.PitchAcc	Vehicle.PitchAcc	Vhcl PitchAcc		rad/s^2	Pitch rotation acceleration
Vhcl.PitchVel	Vehicle.PitchVel	Vhcl PitchVel		rad/s	Pitch rotation speed
Vhcl.PoI.ax_1 Vhcl.PoI.ay_1 Vhcl.PoI.az_1	Vehicle.PoI_Acc_1[0] Vehicle.PoI_Acc_1[1] Vehicle.PoI_Acc_1[2]	Vhcl PoI_Acc_1 x Vhcl PoI_Acc_1 y Vhcl PoI_Acc_1 z	Fr1	m/s^2	Acceleration vector for PoI ("Point of Interest") in vehicle frame (Fr1)
Vhcl.PoI.ax Vhcl.PoI.ay Vhcl.PoI.az	Vehicle.PoI_Acc[0] Vehicle.PoI_Acc[1] Vehicle.PoI_Acc[2]	Vhcl PoI_Acc x Vhcl PoI_Acc y Vhcl PoI_Acc z	Fr0	m/s^2	Acceleration vector for PoI ("Point of Interest") in global frame (Fr0)
Vhcl.PoI.GCS.Elev Vhcl.PoI.GCS.Long Vhcl.PoI.GCS.Lat	Vehicle.PoI_GCS.Elev Vehicle.PoI_GCS.Long Vehicle.PoI_GCS.Lat	Vhcl PoI_GCS Elev Vhcl PoI_GCS Long Vhcl PoI_GCS Lat		m rad rad	Global position for PoI, expressed in GCS coordinates (geodetic elevation, longitude, latitude)
Vhcl.PoI.vx_1 Vhcl.PoI.vy_1 Vhcl.PoI.vz_1	Vehicle.PoI_Vel_1[0] Vehicle.PoI_Vel_1[1] Vehicle.PoI_Vel_1[2]	Vhcl PoI_Vel_1 x Vhcl PoI_Vel_1 y Vhcl PoI_Vel_1 z	Fr1	m/s	Velocity vector for PoI ("Point of Interest") in vehicle frame (Fr1)
Vhcl.PoI.vx Vhcl.PoI.vy Vhcl.PoI.vz	Vehicle.PoI_Vel[0] Vehicle.PoI_Vel[1] Vehicle.PoI_Vel[2]	Vhcl PoI_Vel x Vhcl PoI_Vel y Vhcl PoI_Vel z	Fr0	m/s	Velocity vector for PoI ("Point of Interest") in global frame (Fr0)
Vhcl.PoI.x Vhcl.PoI.y Vhcl.PoI.z	Vehicle.PoI_Pos[0] Vehicle.PoI_Pos[1] Vehicle.PoI_Pos[2]	Vhcl PoI_Pos x Vhcl PoI_Pos y Vhcl PoI_Pos z	Fr0	m	Position for PoI ("Point of Interest") in global frame (Fr0)

Vhcl.<pos>.Fx
Vhcl.<pos>.Fy
Vhcl.<pos>.Fz
Vhcl.<pos>.FxTwin
Vhcl.<pos>.FyTwin
Vhcl.<pos>.FzTwin
	<pre>.Fx
	<pre>.Fy
	<pre>.Fz
	<pre>.FxTwin
	<pre>.FyTwin
	<pre>.FzTwin
		Vhcl Wheel <pos> Fx
		Vhcl Wheel <pos> Fy
		Vhcl Wheel <pos> Fz
			FrW	N	Longitudinal, lateral and vertical ground reaction force at wheel/road contact point <pos>
					(using twin wheel, the first quantity is for the outer and the second is for the inner wheel)

Vhcl.<pos>.LongSlip	<pre>.LongSlip	Vhcl Wheel <pos> LongSlip		-	Longitudinal slip at wheel <pos>

Name UAQ	Name C-Code	Name CM4SL	Frame	Unit	Info (data type, if not double)
Vhcl.<pos>.rot	<pre>.rot	-		rad	Wheel <pos> rotation angle
Vhcl.<pos>.rotv	<pre>.rotv	Vhcl Wheel <pos> rotv		rad/s	Wheel <pos> rotation speed

Vhcl.<pos>.rx
Vhcl.<pos>.ry
Vhcl.<pos>.rz
	<pre>.r_zxy[0]
	<pre>.r_zxy[1]
	<pre>.r_zxy[2]
		Vhcl Wheel <pos> r_zxy x
		Vhcl Wheel <pos> r_zxy y
		Vhcl Wheel <pos> r_zxy z
			Fr1	rad	Rotation angles of carrier <pos> at mounted position in rotation order ZXY

Vhcl.<pos>.SideSlip	<pre>.SideSlip	Vhcl Wheel <pos> SideSlip		rad	Sideslip angle at wheel <pos>
Vhcl.<pos>.Trq_B2WC	<pre>.Trq_B2WC	Vhcl Wheel <pos> Trq_B2WC		Nm	Supported brake torque at wheel carrier <pos>
Vhcl.<pos>.Trq_Brake	<pre>.Trq_Brake	Vhcl Wheel <pos> Trq_Brake		Nm	Total brake torque at wheel <pos>
-	<pre>.Trq_BrakeReg_trg	Vhcl Wheel <pos> Trq_BrakeReg_trg		Nm	Target regenerative brake torque at wheel <pos>
Vhcl.<pos>.Trq_DL2WC	<pre>.Trq_DL2WC	Vhcl Wheel <pos> Trq_DL2WC		Nm	Supported driving torque at wheel carrier <pos>
Vhcl.<pos>.Trq_Drive	<pre>.Trq_Drive	Vhcl Wheel <pos> Trq_Drive		Nm	Total driving torque
Vhcl.<pos>.Trq_T2W	<pre>.Trq_T2W	Vhcl Wheel <pos> Trq_T2W		Nm	Tire torque around wheel spin axle of wheel <pos>
Vhcl.<pos>.Trq_WhlBearing	<pre>.Trq_WhlBearing	Vhcl Wheel <pos> Trq_WhlBearing		Nm	Wheel bearing friction torque around wheel spin axle of wheel <pos>

Vhcl.<pos>.tx
Vhcl.<pos>.ty
Vhcl.<pos>.tz
	<pre>.t[0]
	<pre>.t[1]
	<pre>.t[2]
		Vhcl Wheel <pos> t x
		Vhcl Wheel <pos> t y
		Vhcl Wheel <pos> t z
			Fr1	m	Translation of carrier <pos> at mounted position

Vhcl.<pos>.vBelt	<pre>.vBelt	Vhcl Wheel <pos> vBelt		m/s	Wheel <pos> velocity (based on wheel <pos> rotation speed and effective rolling tire radius)
Vhcl.Roll	Vehicle.Roll	Vhcl Roll		rad	Vehicle roll angle
Vhcl.RollAcc	Vehicle.RollAcc	Vhcl RollAcc		rad/s^2	Vehicle roll acceleration
Vhcl.RollVel	Vehicle.RollVel	Vhcl RollVel		rad/s	Vehicle roll velocity
Vhcl.Road.JuncObjId Vhcl.Road.nextJuncObjId	Vehicle.Road.JuncObjId Vehicle.Road.nextJuncObjId	-			Actual (last) and next road junction object Id
Vhcl.Road.LinkObjId	Vehicle.Road.LinkObjId	-			Actual road link object Id
Vhcl.Road.onJunction	Vehicle.Road.onJunction	-			Flag if the vehicle is on junction?
Vhcl.Road.s2lastJunc Vhcl.Road.s2nextJunc	Vehicle.Road.s2lastJunc Vehicle.Road.s2nextJunc	-		m	Road distance along route (from the middle of the front axle) to last / next junction
Vhcl.tRoad	Vehicle.tRoad	-		m	Lateral distance to route centerline
Vhcl.sRoad	Vehicle.sRoad	Vhcl sRoad		m	Vehicle route / path coordinate
Vhcl.Steer.Acc	Vehicle.Steering.AngAcc	Vhcl Steering AngAcc		rad/s^2	Steering acceleration at steering wheel
Vhcl.Steer.Ang	Vehicle.Steering.Ang	Vhcl Steering Ang		rad	Steering angle at steering wheel
Vhcl.Steer.Trq	Vehicle.Steering.Trq	Vhcl Steering Trq		Nm	Steering torque at steering wheel
Vhcl.Steer.Vel	Vehicle.Steering.AngVel	Vhcl Steering AngVel		rad/s	Steering velocity at steering wheel
Vhcl.v	Vehicle.v	Vhcl v		m/s	Vehicle velocity
Vhcl.Wind.vx Vhcl.Wind.vy Vhcl.Wind.vz	Vehicle.Wind_vel_0[0] Vehicle.Wind_vel_0[1] Vehicle.Wind_vel_0[2]	Vehicle Wind_vel_0 x Vehicle Wind_vel_0 y Vehicle Wind_vel_0 z	Fr0	m/s	Wind velocity vector
Vhcl.Yaw	Vehicle.Yaw	Vhcl Yaw		rad	Vehicle yaw angle
Vhcl.YawAcc	Vehicle.YawAcc	Vhcl YawAcc		rad/s^2	Vehicle yaw acceleration
Vhcl.YawRate	Vehicle.YawRate	Vhcl YawRate		rad/s	Vehicle yaw velocity

24.6 Vehicle Control

24.6.1 General

Name UAQ	Name C-Code	Name CM4SL	Unit	Info (data type, if not double)
<pre>: = VehicleControl				
-	-	VhclCtrl Steering SteerByTrq	-	Flag: Steering by torque is active (boolean)
VC.Brake	<pre>.Brake	VhclCtrl Brake	-	Brake/decelerator activity (0..1)
VC.BrakePark	<pre>.BrakePark	VhclCtrl BrakePark	-	Park brake activity (0..1)
VC.Clutch	<pre>.Clutch	VhclCtrl Clutch	-	Clutch activity (0..1)
VC.Gas	<pre>.Gas	VhclCtrl Gas	-	Gas/throttle/accelerator activity (0..1)
VC.GearNo	<pre>.GearNo	VhclCtrl GearNo	-	Target gear number (integer)
VC.Key	<pre>.Key	VhclCtrl Key	-	Vehicle key position (integer): 0: Key out 1: Key in, power off 2: Key in, power accessory 3: Key in, power on 4: Key in, starter active
VC.Lights.bm	<pre>.Lights.bm	-	-	For internal use only (unsigned short)
VC.Lights.Brake	<pre>.Lights.Brake	VhclCtrl Lights Brake	-	Brake light on (boolean)
VC.Lights.Daytime	<pre>.Lights.Daytime	VhclCtrl Lights Daytime	-	Daytime running light on (boolean)

VC.Lights.FogFrontL
VC.Lights.FogFrontR
	<pre>.Lights.FogFrontL
	<pre>.Lights.FogFrontR
		VhclCtrl Lights FogFrontL
		VhclCtrl Lights FogFrontR
			-	Front fog left/right light on (boolean)

VC.Lights.FogRear	<pre>.Lights.FogRear	VhclCtrl Lights FogRear	-	Rear fog light on (boolean)
VC.Lights.HighBeam	<pre>.Lights.HighBeam	VhclCtrl Lights HighBeam	-	High beam/full headlight on (boolean)

VC.Lights.IndL,
VC.Lights.IndR
	<pre>.Lights.IndL
	<pre>.Lights.IndR
		VhclCtrl Lights IndL
		VhclCtrl Lights IndR
			-	Indicator left/right (integer):
				0=Off; 1=Indicator on, light off; 2=Indicator on, light on

VC.Lights.LowBeam	<pre>.Lights.LowBeam	VhclCtrl Lights LowBeam	-	Low beam/dipped headlight on (boolean)

VC.Lights.ParkL,
VC.Lights.ParkR
	<pre>.Lights.ParkL
	<pre>.Lights.ParkR
		VhclCtrl Lights ParkL
		VhclCtrl Lights ParkR
			-	Parking left/right light on (boolean)

VC.Lights.Reverse	<pre>.Lights.Reverse	VhclCtrl Lights Reverse	-	Reversing light on (boolean)
VC.SelectorCtrl	<pre>.SelectorCtrl	VhclCtrl SelectorCtrl	-	Automatic gear selector position (integer): -9 = position P (corresponds to gear number -9) -1 = position R (corresponds to gear number -1...-8) 0 = position N (corresponds to gear number =0) 1 = position D (corresponds to gear number >0) 2 = position M (manual gear selection / Manumatic)
VC.SST	<pre>.SST	VhclCtrl SST	-	Powertrain start-stop button (boolean): 0=off, 1=on
VC.Steer.Ang	<pre>.Steering.Ang	VhclCtrl Steering Ang	rad	Steering angle at steering wheel
VC.Steer.AngAcc	<pre>.Steering.AngAcc	VhclCtrl Steering AngAcc	rad/s^2	Steering acceleration at steering wheel
VC.Steer.AngVel	<pre>.Steering.AngVel	VhclCtrl Steering AngVel	rad/s	Steering velocity at steering wheel
VC.Steer.Trq	<pre>.Steering.Trq	VhclCtrl Steering Trq	Nm	Steering torque at steering wheel
VC.UserSignal_<i>	<pre>.UserSignal[]	VhclCtrl UserSignal	-	User defined signal <i> from vehicle operator to powertrain control (e.g. seat belt, vehicle door, seat sensor)

24.6.2 ACC-Controller & AccelCtrl

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<pre> := AccelCtrl.ACC_ECU			
AccelCtrl.DesiredAx	AccelCtrl.DesrAx	m/s^2	Desired longitudinal acceleration for the acceleration controller AccelCtrl (value NOTSET=-99999 means no desired acceleration to use)
AccelCtrl.ACC.IsActive	<pre>.IsActive		Flag if the ACC controller is activated
AccelCtrl.ACC.DesiredAx	<pre>.DesrAx	m/s^2	Desired longitudinal acceleration of ACC controller
AccelCtrl.ACC.DesiredDist	<pre>.DesrDist	m	Desired distance to the target vehicle
AccelCtrl.ACC.DesiredSpd	<pre>.DesrSpd	m/s	Desired longitudinal velocity of ACC controller
AccelCtrl.ACC.DesiredTGap	<pre>.DesrTGap	s	Desired time distance to the target vehicle
AccelCtrl.ACC.Time2Collision	<pre>.Time2Collision	s	Time to collision with the target vehicle

24.6.3 Generic Longitudinal Control

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
LongCtrl.AEB.SwitchedOn			Flag if AEB system is switched on
LongCtrl.AEB.IsActive			Flag if AEB system is active (=braking)
LongCtrl.AEB.Target.ObjId			Identification number of relevant target global object: - value (integer) =-1 stands for no detection - value individual to each scenario - value not contiguous, so it is unsuitable for use as an array index
LongCtrl.AEB.Target.Vel		m/s	Longitudinal velocity of relevant target object
LongCtrl.AEB.Target.Decel		m/s^2	Longitudinal deceleration of relevant target object (positive if braking)
LongCtrl.AEB.dDist		m	Relative distance to relevant target object
LongCtrl.AEB.dVel		m/s	Relative velocity to relevant target object
LongCtrl.AEB.dDecel		m/s^2	Relative deceleration to relevant target object
LongCtrl.AEB.Decel_req		m/s^2	Required deceleration to brake to equal velocity of the target object
LongCtrl.AEB.Decel_trg		m/s^2	Target deceleration by AEB system for the acceleration controller
LongCtrl.AEB.Time2Collision		s	Time to collision with the target vehicle
LongCtrl.AEB.Time2Brake		s	Time threshold to brake to equal velocity of the target object
LongCtrl.FCW.SwitchedOn			Flag if FCW system is switched on
LongCtrl.FCW.WarnLevel			Warning level of FCW system: 0=no warning, 1-2=warning level 1-2

24.6.4 Generic Lateral Control

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
LatCtrl.LKAS.SwitchedOn			Flag if LKAS system is switched on
LatCtrl.LKAS.IsActive			Flag if LKAS system is active (steering assist torque)
LatCtrl.LKAS.UsePrioLines			Flag if priority (=yellow) lines are used for lane identification
LatCtrl.LKAS.AssistTrq		Nm	Assist steering torque calculated by LKAS system
LatCtrl.LKAS.AssistTrq_raw		Nm	Raw (unfiltered) assist steering torque
LatCtrl.LKAS.CurveXY_trg		1/m	Target path curvature on xy-plane
LatCtrl.LDW.SwitchedOn			Flag if LDW system is switched on
LatCtrl.LDW.IsReady			Flag if LDW system is ready
LatCtrl.LDW.IsActive			Flag if LDW system is active (warning on)
LatCtrl.LineDetectMode			Specifies the line detection mode: 0=with RPSensor, 1=with LineSensor
LatCtrl.DistToLeft LatCtrl.DistToRight		m	Distance to left/right detected line of the identified lane
LatCtrl.DevDist		m	Deviation distance to path (middle of the lane)
LatCtrl.Lines.ValidPairExists			Flag if a valid pair of priority or standard lines exists
LatCtrl.Lines.<LineId>.exists			Flag if priority/standard line (<LineId>=PrioL, PrioR, StdL, StdR) exists

24.7 Car

24.7.1 Car Convenience Quantities

The following frequently needed quantities are aliases for some of the Car.Con... quantities. The systematic naming convention for other User Accessible Quantities was dropped in favor of ease of use and faster access.

Name UAQ	Name C-Code	Frame	Unit	Info (data type, if not double)
Car.alHori	Car.ConBdy1.alHori	-	m/s^2	Centripetal acceleration (in horizontal plane), Car.alHori is perpendicular to Car.v and to Z-axis (ISO 8855:2011, 5.1.15)
Car.atHori	Car.ConBdy1.atHori	-	m/s^2	Tangential acceleration (in horizontal plane), Car.atHori is parallel to Car.v and perpendicular to Z-axis (ISO 8855:2011, 5.1.14)
Car.ax Car.ay Car.az	Car.ConBdy1.a_1[0] Car.ConBdy1.a_1[1] Car.ConBdy1.a_1[2]	Fr1	m/s^2	Translational acceleration of vehicle connected body without considering gravity in vehicle frame
Car.tx Car.ty Car.tz	Car.ConBdy1.t_0[0] Car.ConBdy1.t_0[1] Car.ConBdy1.t_0[2]	Fr0	m	Translational position of vehicles connected body
Car.v	Car.ConBdy1.v	-	m/s	Absolute velocity of vehicle connected body
Car.vx Car.vy Car.vz	Car.ConBdy1.v_1[0] Car.ConBdy1.v_1[1] Car.ConBdy1.v_1[2]	Fr1	m/s	Translational velocity of vehicle connected body

24.7.2 Car

Name UAQ	Name C-Code	Name CM4SL	Frame	Unit	Info (data type, if not double)
<pos> := FL, FR, RL, RR <part> := Buf, Damp, Spring, Stabi <i> := 0, 1, 2 (q0=wheel compression, q1=opposite wheel compression, q2=rack travel) Twin := used by the quantities for a twin wheel (describes the inner tire) <preLane> := Car.Road.Lane; <preRS> := Car.FARoadSensor [i] := 0,1,2,3,4,5,6,7 (Applicable in case of C-Code, numbers represent FL, FR, RL, RR, FL2, FR2, RL2, RR2 respectively)					
Car.Aero.Frc_1.x Car.Aero.Frc_1.y Car.Aero.Frc_1.z		car Aero Frc_1 x car Aero Frc_1 y car Aero Frc_1 z	Fr1	N	Aerodynamic forces to the vehicle
Car.Aero.tau_1		car Aero tau_1	Fr1	rad	Angle of wind approach direction $\text{Car.Aero.tau} = \text{atan}\frac{-\text{Car.Aero.vyres}}{-\text{Car.Aero.vxres}}$
Car.Aero.tau2_1		-	Fr1	rad	Angle of wind direction in vehicle frame (Fr1) shifted by 2pi Car.Aero.tau_1 + pi
-		car Aero PoA_1 x car Aero PoA_1 y car Aero PoA_1 z	Fr1	m	Attack point for the aero forcesi
Car.Aero.Trq_1.x Car.Aero.Trq_1.y Car.Aero.Trq_1.z		car Aero Trq_1 x car Aero Trq_1 y car Aero Trq_1 z	Fr1	Nm	Aerodynamics torques to vehicle at reference point
Car.Aero.vres_1.x Car.Aero.vres_1.y Car.Aero.vres_1.z		car Aero ApproachVel_1 x car Aero ApproachVel_1 y car Aero ApproachVel_1 z	Fr1	m/s	Relative wind velocity in vehicle frame
Car.Buffer<pos>.Frc Car.Buffer<pos>.Frc_ext Car.Buffer<pos>.Frc_tot		- FBuf <pos> -		N	Buffer force internal Buffer force external Buffer force total (section 11.1 'External Suspension Forces')
Car.Buffer<pos>.Frc_tot.q<i>		-		N	Buffer force due to the generalized coordinate q<i>
Car.Buffer<pos>.l Car.Buffer<pos>.l _com Car.Buffer<pos>.l_ext Car.Buffer<pos>.l_kin		lBuf <pos>	Fr1	m	Buffer length total Buffer length by compliance Buffer length extra, offset Buffer length by kinematics
Car.Buffer<pos>v		vBuf <pos>	Fr1	m/s	Buffer compression/extension velocity
Car.Camber<pos>	Car.Tire[i].Camber	-	Fr1	rad	Camber angle (ISO 8855:2011, 7.1.17). Angle between z-axis of the vehicle and wheel plane. Positive, if the top of the wheel is inclined towards the outside of the vehicle. (Please note the difference between camber angle and inclination angle.)
Car.Con.alHori	Car.ConBdy1.alHori	-		m/s^2	Centripetal acceleration (in horizontal plane), Car.alHori is perpendicular to Car.v and to Z-axis (ISO 8855:2011, 5.1.15)
Car.ConA.Pitch_X			FrX	rad	Vehicle pitch angle of body A
Car.ConA.Roll_X		-	FrX	rad	Vehicle roll angle of body A
Car.Con.atHori	Car.ConBdy1.atHori	-		m/s^2	Tangential acceleration (in horizontal plane), Car.atHori is parallel to Car.v and perpendicular to y-axis (ISO 8855:2011, 5.1.14)
Car.Con.ax_1 Car.Con.ay_1 Car.Con.az_1	Car.ConBdy1.a_1[0] Car.ConBdy1.a_1[1] Car.ConBdy1.a_1[2]	-	Fr1	m/s^2	Translational acceleration of vehicle connected body without considering gravity in vehicle frame
Car.Con.ax Car.Con.ay Car.Con.az	Car.ConBdy1.a_0[0] Car.ConBdy1.a_0[1] Car.ConBdy1.a_0[2]	-	Fr0	m/s^2	Translational acceleration of vehicle connected body without considering gravity in global frame
Car.ConB.Pitch_X			FrX	rad	Vehicle pitch angle of body B
Car.ConB.Roll_X		-	FrX	rad	Vehicle roll angle of body B
Car.Con.tx Car.Con.ty Car.Con.tz	Car.ConBdy1.t_0[0] Car.ConBdy1.t_0[1] Car.ConBdy1.t_0[2]	-	Fr0	m	Translational position of vehicle connected body
Car.Con.v	Car.ConBdy1.v	-		m/s	Velocity of vehicle connected body

Name UAQ	Name C-Code	Name CM4SL	Frame	Unit	Info (data type, if not double)
Car.Con.vHori	Car.ConBdy1.vHori	-		m/s	Horizontal vehicle velocity (ISO 8855:2011, 5.1.5) $Car.vHori = \sqrt{Car.vx^2 + Car.vy^2}$
Car.Con.vx_1 Car.Con.vy_1 Car.Con.vz_1	Car.ConBdy1.v_1[0] Car.ConBdy1.v_1[1] Car.ConBdy1.v_1[2]	-	Fr1	m/s	Translational velocity of vehicle connected body
Car.C<pos>.C.a_0.x Car.C<pos>.C.a_0.y Car.C<pos>.C.a_0.z	Car.Susp[i].WC_a_0[0] Car.Susp[i].WC_a_0[1] Car.Susp[i].WC_a_0[2]	-	Fr0	m/s^2	Acceleration of the carrier center point in Fr0
Car.C<pos>.AxleFrc		-	Fr1	N	Total generalized axle force due to suspension force elements projected on vehicle z-axis
Car.C<pos>.CastorAng Car.C<pos>.CastorOffset		-		rad m	Castor angle and offset at ground (ISO 8855:2011, 7.2.2 and 7.2.3)
Car.C<pos>.C.t_0.x Car.C<pos>.C.t_0.y Car.C<pos>.C.t_0.z		-	Fr0	m	Translation of carrier center point in Fr0
Car.C<pos>.C.v_0.x Car.C<pos>.C.v_0.y Car.C<pos>.C.v_0.z		-	Fr0	m/s	Velocity of the carrier center point in Fr0
Car.C<pos>.Frc2C_1.x Car.C<pos>.Frc2C_1.y Car.C<pos>.Frc2C_1.z		-	Fr1	N	Total force to the wheel carrier in vehicle frame
Car.C<pos>.Frc2C_2.x Car.C<pos>.Frc2C_2.y Car.C<pos>.Frc2C_2.z		-	Fr2	N	Total force to the wheel carrier in carrier frame
Car.C<pos>.Frc2CExt_0.x Car.C<pos>.Frc2CExt_0.y Car.C<pos>.Frc2CExt_0.z	Susp_Ext.Frc2CExt_0[<pos>][0] Susp_Ext.Frc2CExt_0[<pos>][1] Susp_Ext.Frc2CExt_0[<pos>][2]	-	Fr0	N	External forces to the wheel carrier, defined in general frame
Car.C<pos>.Frc2CExt_1.x Car.C<pos>.Frc2CExt_1.y Car.C<pos>.Frc2CExt_1.z	Susp_Ext.Frc2CExt_1[<pos>][0] Susp_Ext.Frc2CExt_1[<pos>][1] Susp_Ext.Frc2CExt_1[<pos>][2]	-	Fr1	N	External forces to the wheel carrier, defined in vehicle frame
Car.C<pos>.Frc2CExt_2.x Car.C<pos>.Frc2CExt_2.y Car.C<pos>.Frc2CExt_2.z	Susp_Ext.Frc2CExt_2[<pos>][0] Susp_Ext.Frc2CExt_2[<pos>][1] Susp_Ext.Frc2CExt_2[<pos>][2]	-	Fr2	N	External forces to the wheel carrier, defined in carrier frame
Car.C<pos>.GenFrc0 Car.C<pos>.GenFrc1 Car.C<pos>.GenFrc2		- - car Susp <pos> GenFrc2		N	Generalized force for the generalized coordinate <i>
Car.C<pos>.GenInert0 Car.C<pos>.GenInert1 Car.C<pos>.GenInert2		- - car Susp <pos> GenInert2		kg	Generalized inertia for the generalized coordinate <i>
Car.C<pos>.KingpinAng Car.C<pos>.KingpinOffset		-		rad m	Kingpin angle and offset at ground (ISO 8855:2011, 7.2.5 and 7.2.6)
Car.C<pos>.l<part>_dq<i>		-	Fr1	m	Change of the length of <part> at <pos> in direction of the generalized coordinate <i> Example: Car.CFL.lSpring_dq0
Car.C<pos>.Maggi0.x Car.C<pos>.Maggi0.y Car.C<pos>.Maggi0.z		-	Fr1		Elements of the "Maggi-Matrix" (describes the relation between the generalized coordinate q0 and the rotation of the wheel carrier)
Car.C<pos>.Maggi1.x Car.C<pos>.Maggi1.y Car.C<pos>.Maggi1.z		-	Fr1		Elements of the "Maggi-Matrix" (describes the relation between the generalized coordinate q1 and the rotation of the wheel carrier)
Car.C<pos>.Maggi2.x Car.C<pos>.Maggi2.y Car.C<pos>.Maggi2.z		-	Fr1		Elements of the "Maggi-Matrix" (describes the relation between the generalized coordinate q2 and the rotation of the wheel carrier)
Car.C<pos>.P.t_0.x Car.C<pos>.P.t_0.y Car.C<pos>.P.t_0.z Car.C<pos>.Twin.P.t_0.x Car.C<pos>.Twin.P.t_0.y Car.C<pos>.Twin.P.t_0.z	Car.Tire[i].P_0[0] Car.Tire[i].P_0[1] Car.Tire[i].P_0[2] Car.TwinTire[i].P_0[0] Car.TwinTire[i].P_0[1] Car.TwinTire[i].P_0[2]	-	Fr0	m	Translation tire contact point in Fr0

Name UAQ	Name C-Code	Name CM4SL	Frame	Unit	Info (data type, if not double)
Car.C<pos>.P.v01_W.x Car.C<pos>.P.v01_W.y Car.C<pos>.P.v01_W.z Car.C<pos>.Twin.P.v01_W.x Car.C<pos>.Twin.P.v01_W.y Car.C<pos>.Twin.P.v01_W.z		-	FrW	m/s	Velocity of the tire contact point in FrW
Car.C<pos>.q<i> Car.C<pos>.q<i>p Car.C<pos>.q<i>pp		-		 m m/s m/s^2	Generalized coordinate q<i> displacement velocity acceleration
Car.C<pos>.rx Car.C<pos>.ry Car.C<pos>.rz	Car.Susp[i].r_zxy[0] Car.Susp[i].r_zxy[1] Car.Susp[i].r_zxy[2]	-	Fr1	rad	Rotation angles of carrier <pos> at wheel center with rotation sequence ZXY
Car.C<pos>.rx_com Car.C<pos>.ry_com Car.C<pos>.rz_com		-	Fr1	rad	Rotation angles of carrier <pos> by compliance with rotation sequence ZYX
Car.C<pos>.rx_dq<i> Car.C<pos>.ry_dq<i> Car.C<pos>.rz_dq<i>		-			Change of the angle at <pos> in direction of the generalized coordinate <i>
Car.C<pos>.rx_ext Car.C<pos>.ry_ext Car.C<pos>.rz_ext	Susp_Ext.Kin_tExt[<pos>][3] Susp_Ext.Kin_tExt[<pos>][4] Susp_Ext.Kin_tExt[<pos>][5]	-	Fr1	rad	Rotation angles of carrier <pos> extra, offset with rotation sequence ZYX
Car.C<pos>.rx_kin Car.C<pos>.ry_kin Car.C<pos>.rz_kin		-	Fr1	rad	Rotation angles of carrier <pos> by kinematics with rotation sequence ZYX
Car.C<pos>.rxv Car.C<pos>.ryv Car.C<pos>.rzv	Susp_Ext.Kin_vExt[<pos>][3] Susp_Ext.Kin_vExt[<pos>][4] Susp_Ext.Kin_vExt[<pos>][5]	-	Fr1	rad/s	Rotation speed of the carrier reference point by kinematics with rotation sequence ZYX
Car.C<pos>.rxv_ext Car.C<pos>.ryv_ext Car.C<pos>.rzv_ext		-	Fr1	rad/s	External rotation speed of the carrier reference point with rotation sequence ZYX (section 12.7 'Additional External Movement')
Car.C<pos>.Tire.FrcC.x Car.C<pos>.Tire.FrcC.y Car.C<pos>.Tire.FrcC.z		-	Fr2	N	Tire force to the wheel carrier
Car.C<pos>.Tire.TrqC.x Car.C<pos>.Tire.TrqC.y Car.C<pos>.Tire.TrqC.z		-	Fr2	Nm	Tire torque to the wheel carrier
Car.C<pos>.Trq2C_1.x Car.C<pos>.Trq2C_1.y Car.C<pos>.Trq2C_1.z		-	Fr1	Nm	Total torque to the wheel carrier in vehicle frame
Car.C<pos>.Trq2C_2.x Car.C<pos>.Trq2C_2.y Car.C<pos>.Trq2C_2.z		-	Fr2	Nm	Total torque to the wheel carrier in carrier frame
Car.C<pos>.Trq2CExt_1.x Car.C<pos>.Trq2CExt_1.y Car.C<pos>.Trq2CExt_1.z	Susp_Ext.Trq2CExt_1[<pos>][0] Susp_Ext.Trq2CExt_1[<pos>][1] Susp_Ext.Trq2CExt_1[<pos>][2]	-	Fr1	Nm	External torques to the wheel carrier in vehicle frame
Car.C<pos>.Trq2CExt_2.x Car.C<pos>.Trq2CExt_2.y Car.C<pos>.Trq2CExt_2.z	Susp_Ext.Trq2CExt_2[<pos>][0] Susp_Ext.Trq2CExt_2[<pos>][1] Susp_Ext.Trq2CExt_2[<pos>][2]	-	Fr2	Nm	External torques to the wheel carrier in carrier frame
Car.C<pos>.Trq_B2C				Nm	Brake torque at suspension <pos> after reduction
Car.C<pos>.TrqGyro_2.x Car.C<pos>.TrqGyro_2.z		-	Fr2	Nm	Gyroscopic torque at wheel carrier <pos>
Car.C<pos>.Trq_T2W Car.C<pos>.Twin.Trq_T2W		Car Trq_T2W<pos>		Nm	Tire torque around wheel spin axis
Car.C<pos>.tx Car.C<pos>.ty Car.C<pos>.tz	Car.Susp[i].t[0] Car.Susp[i].t[1] Car.Susp[i].t[2]	-	Fr1	m	Translation of carrier reference point at <pos>
Car.C<pos>.tx_com Car.C<pos>.ty_com Car.C<pos>.tz_com		-	Fr1	m	Translation of carrier reference point by compliance
Car.C<pos>.tx_dq<i> Car.C<pos>.ty_dq<i> Car.C<pos>.tz_dq<i>		-	Fr1	m	Translation of wheel carrier at <pos> in direction of the generalized coordinate <i> Example: Car.CFL.tz_dq0
Car.C<pos>.tx_ext Car.C<pos>.ty_ext Car.C<pos>.tz_ext	Susp_Ext.Kin_tExt[<pos>][0] Susp_Ext.Kin_tExt[<pos>][1] Susp_Ext.Kin_tExt[<pos>][2]	-	Fr1	m	Translation of carrier reference point extra, offset (section 12.7 'Additional External Movement')
Car.C<pos>.tx_kin Car.C<pos>.ty_kin Car.C<pos>.tz_kin		-	Fr1	m	Translation of carrier reference point by kinematics

Name UAQ	Name C-Code	Name CM4SL	Frame	Unit	Info (data type, if not double)
Car.C<pos>.txv Car.C<pos>.tyv Car.C<pos>.tzv		-	Fr1	m/s	Velocity of the carrier reference point by kinematics
Car.C<pos>.txv_ext Car.C<pos>.tyv_ext Car.C<pos>.tzv_ext	Susp_Ext.Kin_vExt[<pos>][0] Susp_Ext.Kin_vExt[<pos>][1] Susp_Ext.Kin_vExt[<pos>][2]	-	Fr1	m/s	Additional external velocity (user defined) of the carrier reference point (section 12.7 'Additional External Movement')
Car.Damp<pos>.Frc Car.Damp<pos>.Frc_ext Car.Damp<pos>.Frc_tot		- FDamp <pos> -		N	Damper force internal Damper force external Damper force total (section 11.1 'External Suspension Forces')
Car.Damp<pos>.Frc_tot.q<i>		-		N	Damper force due to the generalized coordinate q<i>
Car.Damp<pos>.l Car.Damp<pos>.l_com Car.Damp<pos>.l_ext Car.Damp<pos>.l_kin		lDamp <pos>		m	Damper length total Damper length by compliance Damper length extra, offset Damper length by kinematics
Car.Damp<pos>.TopMnt.l		-		m	Top mount length
Car.Damp<pos>.TopMnt.v		-		m/s	Top mount compression/extension velocity
Car.Damp<pos>.v		vDamp <pos>		m/s	Damper compression/extension velocity
Car.Distance	Car.Distance	-	Fr0	m	Driving distance since TestRun start $\text{Car.Distance} = \int_t \text{Car.v}\, dt$
Car.Fr1.ax Car.Fr1.ay Car.Fr1.az	Car.Fr1.a_0[0] Car.Fr1.a_0[1] Car.Fr1.a_0[2]	-	Fr0	m/s^2	Acceleration of origin Fr1 in global coordinate system
Car.Fr1.rx Car.Fr1.ry Car.Fr1.rz	Car.Fr1.r_zyx[0] Car.Fr1.r_zyx[1] Car.Fr1.r_zyx[2]	Car Fr1 rx Car Fr1 ry Car Fr1 rz	Fr0	rad	Rotation angles of origin Fr1 in global coordinate system, Cardan angles with rotation sequence ZYX (ISO 8855:2011, 5.2.1, 5.2.2 and 5.2.3)
Car.Fr1.tx Car.Fr1.ty Car.Fr1.tz	Car.Fr1.t_0[0] Car.Fr1.t_0[1] Car.Fr1.t_0[2]	-	Fr0	m	Position of origin Fr1 in global coordinate system
Car.Fr1.vx Car.Fr1.vy Car.Fr1.vz	Car.Fr1.v_0[0] Car.Fr1.v_0[1] Car.Fr1.v_0[2]	-	Fr0	m/s	Translational velocity of origin Fr1 in global coordinate system
Car.Fx<pos> Car.Fy<pos> Car.Fz<pos> Car.FxTwin<pos> Car.FyTwin<pos> Car.FzTwin<pos>	Car.Tire[i].Frc_W[0] Car.Tire[i].Frc_W[1] Car.Tire[i].Frc_W[2] Car.TwinTire[i].Frc_W[0] Car.TwinTire[i].Frc_W[1] Car.TwinTire[i].Frc_W[2]	-	FrW	N	Longitudinal, lateral and vertical ground reaction force at wheel/road contact point
Car.Gen.ax_1 Car.Gen.ay_1 Car.Gen.az_1	Car.GenBdy1.a_1[0] Car.GenBdy1.a_1[1] Car.GenBdy1.a_1[2]	-	Fr1	m/s^2	Acceleration of generalized vehicle body in vehicle frame without considering gravity
Car.Gen.ax Car.Gen.ay Car.Gen.az	Car.GenBdy1.a_0[0] Car.GenBdy1.a_0[1] Car.GenBdy1.a_0[2]	-	Fr0	m/s^2	Acceleration of generalized vehicle body in global frame without considering gravity
Car.Gen.tx Car.Gen.ty Car.Gen.tz	Car.GenBdy1.t_0[0] Car.GenBdy1.t_0[1] Car.GenBdy1.t_0[2]	-	Fr0	m	Position of generalized vehicle body
Car.Gen.vx_1 Car.Gen.vy_1 Car.Gen.vz_1	Car.GenBdy1.v_1[0] Car.GenBdy1.v_1[1] Car.GenBdy1.v_1[2]	-	Fr1	m/s	Velocity of generalized vehicle body
Car.Hitch.Frc2Car.x Car.Hitch.Frc2Car.y Car.Hitch.Frc2Car.z	Car.Hitch.Frc2Car_0[0] Car.Hitch.Frc2Car_0[1] Car.Hitch.Frc2Car_0[2]	Car Hitch Frc2Car x Car Hitch Frc2Car y Car Hitch Frc2Car z	Fr0	N	Trailer hitch force acting on car
Car.Hitch.Trq2Car.x Car.Hitch.Trq2Car.y Car.Hitch.Trq2Car.z	Car.Hitch.Trq2Car_0[0] Car.Hitch.Trq2Car_0[1] Car.Hitch.Trq2Car_0[2]	Car Hitch Trq2Car x Car Hitch Trq2Car y Car Hitch Trq2Car z		Nm	Trailer hitch torque acting on car
Car.Hitch.tx Car.Hitch.ty Car.Hitch.tz	Car.Hitch.t_0[0] Car.Hitch.t_0[1] Car.Hitch.t_0[2]	Car Hitch tx Car Hitch ty Car Hitch tz	Fr0	m	Trailer hitch position (hitch center reference point) in global frame
Car.Hitch.vx Car.Hitch.vy Car.Hitch.vz	Car.Hitch.v_0[0] Car.Hitch.v_0[1] Car.Hitch.v_0[2]	Car Hitch vx Car Hitch vy Car Hitch vz	Fr0	m/s	Trailer hitch velocity (hitch center reference point)

Name UAQ	Name C-Code	Name CM4SL	Frame	Unit	Info (data type, if not double)
Car.InclinAngle<pos> Car.InclinAngleTwin<pos>	Car.Tire[i].InclinAngle	-	FrW	rad	Wheel inclination angle (ISO 8855:2011, 7.1.16). Angle between z-axis of the wheel axis system and wheel plane. Positive for positive rotation around x-axis of the wheel
Car.Jack.Fz<pos>		-	Fr0	N	Externally applied force from jack to vehicle body (component in z-direction)
Car.Jack.tz<pos>	Car.Jack[i].tz	-	Fr0	m	Externally applied jack translation along the z-axis
Car.KinPitchCenter_L.x_1 Car.KinPitchCenter_L.y_1 Car.KinPitchCenter_L.z_1		-	Fr1	m	Kinematical pitch center on the plane through the left side wheel contact points, expressed in vehicle frame (section 12.5.2 'Kinematic characteristic parameters')
Car.KinPitchCenter_L.x Car.KinPitchCenter_L.z		-		m	Kinematical pitch center on the plane through the left side wheel contact points, expressed in plane coordinates (section 12.5.2 'Kinematic characteristic parameters')
Car.KinPitchCenter_R.x_1 Car.KinPitchCenter_R.y_1 Car.KinPitchCenter_R.z_1		-	Fr1	m	Kinematical pitch center on the plane through the right side wheel contact points, expressed in vehicle frame
Car.KinPitchCenter_R.x Car.KinPitchCenter_R.z		-		m	Kinematical pitch center on the plane through the right side wheel contact points, expressed in plane coordinates (section 12.5.2 'Kinematic characteristic parameters')
Car.KinRollCenter_F.x_1 Car.KinRollCenter_F.y_1 Car.KinRollCenter_F.z_1		-	Fr1	m	Kinematical roll center on the front transverse plane through the wheel contact points, expressed in vehicle frame
Car.KinRollCenter_F.y Car.KinRollCenter_F.z		-		m	Kinematical roll center on the front transverse plane through the wheel contact points, expressed in plane coordinates (section 12.5.2 'Kinematic characteristic parameters')
Car.KinRollCenter_R.x_1 Car.KinRollCenter_R.y_1 Car.KinRollCenter_R.z_1		-	Fr1	m	Kinematical roll center on the rear transverse plane through the wheel contact points, expressed in vehicle frame
Car.KinRollCenter_R.y Car.KinRollCenter_R.z		-		m	Kinematical roll center on the rear transverse plane through the wheel contact points, expressed in plane coordinates (section 12.5.2 'Kinematic characteristic parameters')
Car.Load.<i>.mass	Car.Load[i].mass	Car Load_<i> mass		kg	Mass of car load <i>
Car.Load.<i>.tx Car.Load.<i>.ty Car.Load.<i>.tz	Car.Load[i].pos[0] Car.Load[i].pos[1] Car.Load[i].pos[2]	Car Load_<i> tx Car Load_<i> ty Car Load_<i> tz	Fr1	m	Position of car load <i> in vehicle frame
Car.LongSlip<pos> Car.LongSlipTwin<pos>	Car.Tire[i].LongSlip	-		–	Longitudinal slip. Definition depends on tire model (section 17.1.1 'Tire Model with Contact Point Interface (CPI)')
Car.muRoad<pos> Car.muRoadTwin<pos>	Car.Tire[i].muRoad	-		–	Road friction coefficient at tire <pos>
Car.Pitch	Car.Pitch	-		rad	Vehicle pitch angle (ISO 8855:2011, 5.2.2) Positive, if front goes down and rear of the car comes up
Car.PitchAcc	Car.PitchAcc	-		rad/s^2	Pitch rotation acceleration
Car.PitchVel	Car.PitchVel	Car Fr1 rvy		rad/s	Pitch rotation velocity
Car.Road.GCS.Long Car.Road.GCS.Lat Car.Road.GCS.Elev	<preRS>.P_GCS.Long <preRS>.P_GCS.Lat <preRS>.P_GCS.Elev	-	GCS	rad rad m	Global position of the road reference point (in the middle of the front axle) in GCS coordinates: longitude, latitude, elevation
Car.Road.JuncObjId Car.Road.nextJuncObjId	<preRS>.JuncObjId <preRS>.nextJuncObjId				Actual (last) and next road junction object Id
<preLane>.Act.isRight <preLane>.OnLeft.isRight <preLane>.OnRight.isRight	<preRS>.Act.isRight <preRS>.OnLeft.isRight <preRS>.OnRight.isRight	-		-	Indicates if the actual lane, at which the road sensor reference point (in the middle of the front axle) is, is on right side along route (same for lanes on left / right side)
<preLane>.Act.LaneId <preLane>.OnLeft.LaneId <preLane>.OnRight.LaneId	<preRS>.Act.LaneId <preRS>.OnLeft.LaneId <preRS>.OnRight.LaneId	-		-	Indicates the lane Id at which the road sensor reference point (in the middle of the front axle) is, same for lanes on left / right side (-1 means no lane found)
<preLane>.Act.tMidLane <preLane>.OnLeft.tMidLane <preLane>.OnRight.tMidLane	<preRS>.Act.tMidLane <preRS>.OnLeft.tMidLane <preRS>.OnRight.tMidLane	-		m	Lateral offset of lane mid to the route center line

Name UAQ	Name C-Code	Name CM4SL	Frame	Unit	Info (data type, if not double)
<preLane>.Act.Type <preLane>.OnLeft.Type <preLane>.OnRight.Type	<preRS>.Act.Type <preRS>.OnLeft.Type <preRS>.OnRight.Type	-		-	Type of the actual lane (same for lanes on left / right side): 0: legally driveable road 1: bicycle lane 2: lane only for pedestrian 3: traffic island 4: border lane 5: roadside
<preLane>.Act.Width <preLane>.OnLeft.Width <preLane>.OnRight.Width	<preRS>.Act.Width <preRS>.OnLeft.Width <preRS>.OnRight.Width	-		m	Width of the actual roadway lane and the lanes on left / right side
Car.Road.Lane.nLeft Car.Road.Lane.nRight	<preRS>.Lanes.nLanesL <preRS>.Lanes.nLanesR	-		-	Number of left, right roadway lanes along route (integer)
Car.Road.LinkObjId	<preRS>.LinkObjId				Actual road link object Id
Car.Road.onJunction	<preRS>.onJunction				Flag if the vehicle is on junction?
Car.Road.Path.DevAng Car.Road.Route.DevAng	<preRS>.Path.Deviation.Ang <preRS>.Route.Deviation.Ang	-		rad	Deviation angle between vehicle and path / route center line in the road sensor reference point (in the middle of the front axle) (section 'Previewed road point along route centerline')
Car.Road.Path.DevDist Car.Road.Route.DevDist	<preRS>.Path.Deviation.Dist <preRS>.Route.Deviation.Dist	-		m	Deviation distance between vehicle and path / route center line in the road sensor reference point (in the middle of the front axle) (section 'Previewed road point along route centerline')
Car.Road.Path.LatSlope Car.Road.Route.LatSlope	<preRS>.Path.LatSlope <preRS>.Route.LatSlope	-		rad	Lateral road slope at road sensor reference point in the middle of the front axle along path / route
Car.Road.Path.LongSlope Car.Road.Route.LongSlope	<preRS>.Path.LongSlope <preRS>.Route.LongSlope	-		rad	Longitudinal road slope at road sensor reference point (in the middle of the front axle) along path / route
Car.Road.s2lastJunc Car.Road.s2nextJunc	<preRS>.s2lastJunc <preRS>.s2nextJunc	-		m	Road distance along route (from the middle of the front axle) to last / next junction
Car.Road.sRoad	Car.sRoad	-		m	Vehicle road coordinate in the road sensor reference point (in the middle of the front axle) along route / path
Car.Road.tx Car.Road.ty Car.Road.tz	Car.RoadSensor.P_0[0] Car.RoadSensor.P_0[1] Car.RoadSensor.P_0[2]	-	Fr0	m	Position of the road sensor reference point (in the middle of the front axle) in global frame
Car.Roll	Car.Roll	-	Fr0	rad	Vehicle roll angle (ISO 8855:2011, 5.2.3) Positive, if right side goes down and left side comes up
Car.RollAcc	Car.RollAcc	-		rad/s^2	Roll rotation acceleration
Car.RollKin_F Car.RollKin_R	Car.RollKin_F Car.RollKin_R	-		rad	Kinematic roll angle (ISO 8855:2011, 5.2.5), angle between front/rear axle and z-axis of vehicle frame Positive, if right side goes down and left side comes up
Car.RollVel	Car.RollVel	Car Fr1 rvx		rad/s	Roll rotation speed
Car.SecSpring<pos>.Frc_tot		-		N	Secondary spring force
Car.SecSpring<pos>.Frc_tot.q<i>		-		N	Generalized force for the coordinate <i> due to the secondary spring force
Car.SideSlipAngle	Car.ConBdy1.SideSlipAngle	-		rad	Sideslip angle (ISO 8855:2011, 5.2.9) Angle between x-axis of the car and direction of Car.vHori $\text{Car.SideSlipAngle} = \text{atan}\frac{\text{Car.vy}}{\text{Car.vx}}$ $\text{Car.SideSlipAngle} = 0$ $\text{if Car.vx} = \text{Car.vy} = 0$ value range -pi .. +pi
Car.SideSlipAngle2	Car.ConBdy1.SideSlipAngle2	-		rad	Sideslip angle, value range 0 ... 2 pi
Car.SideSlipAngleVel	Car.ConBdy1.SideSlipAngleVel	-		rad/s	Sideslip angle velocity (ISO 8855:2011, 5.2.9)
Car.SimPhase	Car.SimPhase	-		–	Car simulation phase

Name UAQ	Name C-Code	Name CM4SL	Frame	Unit	Info (data type, if not double)
Car.SlipAngle<pos> Car.SlipAngleTwin<pos>	Car.Tire[i].SlipAngle	-		rad	Slip angle (ISO 8855:2011, 5.2.12) Angle between x-axis of the wheel and the tangent of the trajectory of the center of tire contact. Positive for positive rotation around z-axis. $\text{Car.SlipAngleFL} = \text{atan}\frac{\text{Car.vyFL}}{\text{Car.vxFL}}$ $\text{Car.SlipAngleFL} = 0$ if Car.vxFL = Car.vyFL = 0
Car.SlotCar.Deviation	Car.SlotCar.Deviation	-		m	Tolerated deviation along road centerline within SlotCar Mode is disabled
Car.SlotCar.State		-		-	Specifies if SlotCar correction is active (boolean) (section 23.2 'SlotCar Mode')
Car.Spring<pos>.Frc Car.Spring<pos>.Frc_ext Car.Spring<pos>.Frc_tot		- FSpring <pos> -		N	Spring force internal Spring force external Spring force total (section 11.1 'External Suspension Forces')
Car.Spring<pos>.Frc_tot.q<i>		-		N	Spring force due to the generalized coordinate q<i>
Car.Spring<pos>.l Car.Spring<pos>.l _com Car.Spring<pos>.l _ext Car.Spring<pos>.l _kin		lSpring <pos> - - -		m	Spring length total Spring length by compliance Spring length extra, offset Spring length by kinematics
Car.Spring<pos>v		vSpring <pos>		m/s	Spring compression/extension velocity
Car.Stabi<pos>.Frc Car.Stabi<pos>.Frc_ext Car.Stabi<pos>.Frc_tot		- FStabi <pos> -		N	Stabilizer force internal Stabilizer force external Stabilizer force total (section 11.1 'External Suspension Forces')
Car.Stabi<pos>.Frc_tot.q<i>		-			Stabilizer force due to the generalized coordinate q<i>
Car.Stabi<pos>.l Car.Stabi<pos>.l_com Car.Stabi<pos>.l_ext Car.Stabi<pos>.l_kin		lStabi <pos> - - -		m	Stabilizer length total Stabilizer length by compliance Stabilizer length extra, offset Stabilizer length by kinematics
Car.Stabi<pos>.v		vStabi <pos>		m/s	Stabilizer compression/extension velocity
Car.StandbyMode		-		-	Car standby mode (integer) (section 23.1 'Standby Mode')
Car.SteerAngle<pos>	Car.Susp[i].SteerAngle	-		rad	Steer angle (ISO 8855:2011, 7.1.1)
Car.Toe<pos> Car.ToeTwin<pos>	Car.Susp[i].Toe	-		rad	Toe angle (ISO 8855:2011, 7.1.6). The toe angle is positive if the front section of the wheel is turned toward the vehicle's longitudinal center plane and is negative (toe-out) if the front section is turned away from this plane.
Car.TrackCurv	Car.ConBdy1.TrackCurv	-		1/m	Curvature of trajectory (ISO 8855:2011, 5.3.5) (float) Car.TrackCurv = 1 / Car.TrackRadius Car.TrackCurv = 0 if Car.TrackRadius = ∞ Car.TrackCurv = ∞ if Car.TrackRadius = 0
Car.TrackRadius	Car.ConBdy1.TrackRadius	-		m	Radius of path/trajectory (ISO 8855:2011, 5.3.3) (float). Distance between a point of the trajectory and the belonging instantaneous center. $\text{Car.TrackRadius} = \frac{(\text{Car.vHori})^2}{\text{Car.alHori}}$ $\text{Car.TrackRadius} = \infty$ if Car.alHori = 0

Name UAQ	Name C-Code	Name CM4SL	Frame	Unit	Info (data type, if not double)
Car.TrqAlign<pos> Car.TrqAlignTwin<pos>	Car.Tire[i].Trq_W[2]	-	FrW	Nm	Aligning torque (ISO 8855:2011, 10.2.13). Component of the ground reaction moment. Positive for positive direction around z-axis
Car.TrqOvert<pos> Car.TrqOvertTwin<pos>	Car.Tire[i].Trq_W[0]	-	FrW	Nm	Overturning torque in the tire contact point. Positive for positive direction around x-axis
Car.TrqRoll<pos> Car.TrqRollTwin<pos>	Car.Tire[i]..Trq_W[1]	-	FrW	Nm	Rolling torque in the tire contact point. Positive for positive direction around y-axis
Car.TurnSlip<pos> Car.TurnSlipTwin<pos	Car.Tire[i].TurnSlip	-		1/m	Turn slip
Car.Virtual.Frc_0.x Car.Virtual.Frc_0.y Car.Virtual.Frc_0.z	Car.Virtual.Frc_0[0] Car.Virtual.Frc_0[1] Car.Virtual.Frc_0[2]	Car Virtual Frc_0 x Car Virtual Frc_0 y Car Virtual Frc_0 z	Fr0	N	Virtual force, defined in Fr0
Car.Virtual.Frc_1.x Car.Virtual.Frc_1.y Car.Virtual.Frc_1.z	Car.Virtual.Frc_1[0] Car.Virtual.Frc_1[1] Car.Virtual.Frc_1[2]	Car Virtual Frc_1 x Car Virtual Frc_1 y Car Virtual Frc_1 z	Fr1	N	Virtual force, defined in Fr1
Car.Virtual.Trq_0.x Car.Virtual.Trq_0.y Car.Virtual.Trq_0.z	Car.Virtual.Trq_0[0] Car.Virtual.Trq_0[1] Car.Virtual.Trq_0[2]	Car Virtual Trq_0 x Car Virtual Trq_0 y Car Virtual Trq_0 z	Fr0	Nm	Virtual torque, defined in Fr0
Car.Virtual.Trq_1.x Car.Virtual.Trq_1.y Car.Virtual.Trq_1.z	Car.Virtual.Trq_1[0] Car.Virtual.Trq_1[1] Car.Virtual.Trq_1[2]	Car Virtual Trq_1 x Car Virtual Trq_1 y Car Virtual Trq_1 z	Fr1	Nm	Virtual torque, defined in Fr1
Car.v<pos> Car.vTwin<pos>	Car.Tire[i].v	-		m/s	Wheel velocity (based on wheel rotation and effective rolling radius) $\text{Car.vFL} = \text{Car.WheelSpd_FL} \times \text{Car.WFL_KinRollRadius}$
Car.vx<pos> Car.vxTwin<pos>	Car.Tire[i].P_vel_W[0]	-	FrW	m/s	Longitudinal velocity of the tire contact point
Car.vy<pos> Car.vyTwin<pos>	Car.Tire[i].P_vel_W[1]	-	FrW	m/s	Lateral velocity of the tire contact point
Car.vz<pos>	Car.Tire[i].P_vel_W[2]	-		m/s	Vertical velocity of the tire contact point
Car.WheelSpd_<pos> Car.WheelSpd_Twin<pos>	Car.Tire[i].WheelSpd	-		rad/s	Wheel rotation speed
Car.WheelTurnSpd_<pos>	Car.Tire[i].WheelTurnSpd	-		rad/s	Wheel turning (yaw) velocity
Car.W<pos>.Radius Car.W<pos>.Twin.Radius	Car.Tire[i].WRadius	-		m	Current wheel radius (distance road to wheel center)
Car.W<pos>.rot	Car.Tire[i].rot	-		rad	Wheel rotation angle
Car.Yaw	Car.Yaw	-		rad	vehicle yaw angle (ISO 8855:2011, 5.2.1) Angle between x-axis of the car and x-axis of earth fixed system. Positive for positive rotation around z-axis.
Car.YawAcc	Car.YawAcc	-		rad/s^2	Vehicle yaw acceleration
Car.YawRate	Car.YawRate	Car Fr1 rvz		rad/s	Vehicle yaw velocity (ISO 8855:2011, 5.2.19)
Car.YawVel	Car.YawRate	-		rad/s	Vehicle yaw velocity (ISO 8855:2011, 5.2.19)

24.7.3 CarFlex

Name UAQ	Frame	Unit	Info (data type, if not double)
Car.ConA.Pitch_X	FrX	rad	Pitch angle of Fr1A
Car.ConA.Roll_X	FrX	rad	Roll angle of Fr1A
Car.ConB.Pitch_X	FrX	rad	Pitch angle of Fr1B
Car.ConB.Roll_X	FrX	rad	Roll angle of Fr1B
Car.Fr1B.rax Car.Fr1B.ray	–	rad/s^2	Rotational accelerations of body flexibility joint
Car.Fr1B.rvx Car.Fr1B.rvy	–	rad/s	Rotational velocity of body flexibility joint
Car.Fr1B.rx Car.Fr1B.ry	–	rad	Rotation angles of body flexibility joint
Car.Fr1B.Trq_B2A.x Car.Fr1B.Trq_B2A.y	Fr1A	Nm	Spring-damper torque of body flexibility joint from Fr1B to Fr1A
Car.Fr1B.Trq_B2A.x_ext Car.Fr1B.Trq_B2A.y_ext	Fr1A	Nm	External torque from Fr1B to Fr1A
FrX is a frame, defined by the tire road contact points			

24.7.4 Elastic mounted Engine

Name UAQ	Name C-Code	Frame	Unit	Info (data type, if not double)
Car.Eng.ax Car.Eng.ay Car.Eng.az	Car.FrEng.a[0] Car.FrEng.a[1] Car.FrEng.a[2]	Fr1A	m/s^2	Translational acceleration of the engine along x,y,z-axle
Car.Eng.Frc.<i>.Frc.x Car.Eng.Frc.<i>.Frc.y Car.Eng.Frc.<i>.Frc.z		Fr1A	N	Total spring-damper force of the mounting <i> (i=0...3) in direction x,y,z
Car.Eng.Frc.<i>.lx Car.Eng.Frc.<i>.ly Car.Eng.Frc.<i>.lz		Fr1A	m	Spring length of the mounting <i> in x,y,z
Car.Eng.Frc.<i>.vx Car.Eng.Frc.<i>.vy Car.Eng.Frc.<i>.vz		Fr1A	m/s	Damper velocity of the mounting <i>
Car.Eng.GenFrc.x Car.Eng.GenFrc.y Car.Eng.GenFrc.z		Fr1A	N	Spring-damper force along x,y,z of the joint between engine and body A
Car.Eng.GenTrq.x Car.Eng.GenTrq.y		Fr1A	Nm	Total spring-damper torque around x- and y-axle in the joint between engine and body A
Car.Eng.rax Car.Eng.ray	Car.FrEng.ra_xy[0] Car.FrEng.ra_xy[1]	FrEng	rad/s^2	Rotational acceleration of the engine around x-axle and y-axle of FrEng
Car.Eng.rvx Car.Eng.rvy	Car.FrEng.rv_xy[0] Car.FrEng.rv_xy[1]	FrEng	rad/s	Rotational speed of the engine around x-axle and y-axle of FrEng
Car.Eng.rx Car.Eng.ry	Car.FrEng.r_xy[0] Car.FrEng.r_xy[1]	FrEng	rad	Rotation of the engine around x-axle and y-axle of FrEng
Car.Eng.tx Car.Eng.ty Car.Eng.tz	Car.FrEng.t[0] Car.FrEng.t[1] Car.FrEng.t[2]	Fr1A	m	Translation of the engine along x,y,z-axle
Car.Eng.vx Car.Eng.vy Car.Eng.vz	Car.FrEng.v[0] Car.FrEng.v[1] Car.FrEng.v[2]	Fr1A	m/s	Translational velocity of the engine along x,y,z-axle
FrEng is the frame of the elastic supported vehicle body containing engine				

24.8 MBS Suspension

24.8.1 Bushing

Following quantities describe the quantities, which are declared for each bushing component:

Name UAQ	Unit	Info (data type, if not double)
<BushingPre>.tx <BushingPre>.ty <BushingPre>.tz	m	Relative compression along the bushing x,y,z- axis
<BushingPre>.vx <BushingPre>.vy <BushingPre>.vz	m/s	Relative compression velocity along the bushing x,y,z- axis
<BushingPre>.rx <BushingPre>.ry <BushingPre>.rz	rad	Relative rotational compression around the bushing x,y,z- axis
<BushingPre>.Frc.x <BushingPre>.Frc.y <BushingPre>.Frc.z	N	Bushing force along the bushing x,y,z- axis
<BushingPre>.Trq.x <BushingPre>.Trq.y <BushingPre>.Trq.z	Nm	Bushing torque around the bushing x,y,z- axis

24.8.2 McPherson suspension

Name UAQ	Unit	Info (data type, if not double)
<pre> := McPherson.<pos> <pos> := SuspF		

<pre>.DamperL.t
<pre>.DamperL.v
<pre>.DamperL.a
	m
	m/s
	m/s^2
		Translation of the left damper relative to the wheel carrier along the thrust damper axis

<pre>.DamperR.t
<pre>.DamperR.v
<pre>.DamperR.a
	m
	m/s
	m/s^2
		Translation of the right damper relative to the wheel carrier along the thrust damper axis

<pre>.OSRate	-	Model oversampling rate (integer)
<pre>.PlungL.Frc2Wheel	N	Normal force at the left tire contact point
<pre>.PlungR.Frc2Wheel	N	Normal force at the right tire contact point
<pre>.StbBarL.ra_y	rad/s^2	Rotation acceleration of the left stabilizer bar body relative to vehicle chassis around the y-axis
<pre>.StbBarL.rv_y	rad/s	Rotation velocity of the left stabilizer bar body relative to vehicle chassis around the y-axis
<pre>.StbBarL.r_y	rad	Rotation of the left stabilizer bar body relative to vehicle chassis around the y-axis
<pre>.StbBarR.ra_y	rad/s^2	Rotation acceleration of the right stabilizer bar body relative to vehicle chassis around the y-axis
<pre>.StbBarR.rv_y	rad/s	Rotation velocity of the right stabilizer bar body relative to vehicle chassis around the y-axis
<pre>.StbBarR.r_y	rad	Rotation of the right stabilizer bar body relative to vehicle chassis around the y-axis
<pre>.StbLinkL.Bush2WC.*		Parameters of the bushing between the left stabilizer link and wheel carrier body
<pre>.StbLinkR.Bush2WC.*		Parameters of the bushing between the right stabilizer link and wheel carrier body

<pre>.StbLinkL.ra_yx.x
<pre>.StbLinkL.ra_yx.y
	rad/s^2	Rotation acceleration of the left stabilizer link body relative to stabilizer bar around the x,y-axis with rotation sequence YX

<pre>.StbLinkL.rv_yx.x
<pre>.StbLinkL.rv_yx.y
	rad/s	Rotation velocity of the left stabilizer link body relative to stabilizer bar around the x,y-axis with rotation sequence YX

<pre>.StbLinkL.r_yx.x
<pre>.StbLinkL.r_yx.y
	rad	Rotation of the left stabilizer link body relative to stabilizer bar around the x,y-axis with rotation sequence YX

<pre>.StbLinkR.ra_yx.x
<pre>.StbLinkR.ra_yx.y
	rad/s^2	Rotation acceleration of the right stabilizer link body relative to stabilizer bar around the x,y-axis with rotation sequence YX

<pre>.StbLinkR.rv_yx.x
<pre>.StbLinkR.rv_yx.y
	rad/s	Rotation velocity of the right stabilizer link body relative to stabilizer bar around the x,y-axis with rotation sequence YX

<pre>.StbLinkR.r_yx.x
<pre>.StbLinkR.r_yx.y
	rad	Rotation of the right stabilizer link body relative to stabilizer bar around the x,y-axis with rotation sequence YX

<pre>.StrRack.a_y	m/s^2	Translation acceleration of the steering rack body relative to the vehicle chassis along the y-axis (information imposed by steering model)
<pre>.StrRack.t_y	m	Translation of the steering rack body relative to the vehicle chassis along the y-axis (information imposed by steering model)
<pre>.StrRack.v_y	m/s	Translation velocity of the steering rack body relative to the vehicle chassis along the y-axis (information imposed by steering model)
<pre>.StrRodL.Bush2StrRack.*		Parameters of the bushing between the left steering rod and steering rack body
<pre>.StrRodR.Bush2StrRack.*		Parameters of the bushing between the right steering rod and steering rack body

<pre>.StrRodL.ra_zx.x
<pre>.StrRodL.ra_zx.z
	rad/s^2	Rotation acceleration of the left steering rod relative to the wheel carrier body around the x,z-axis with rotation sequence ZX

<pre>.StrRodL.rv_zx.x
<pre>.StrRodL.rv_zx.z
	rad/s	Rotation velocity of the left steering rod relative to the wheel carrier body around the x,z-axis with rotation sequence ZX

<pre>.StrRodL.r_zx.x
<pre>.StrRodL.r_zx.z
	rad	Rotation of the left steering rod relative to the wheel carrier body around the x,z-axis with rotation sequence ZX

<pre>.StrRodR.ra_zx.x
<pre>.StrRodR.ra_zx.z
	rad/s^2	Rotation acceleration of the right steering rod relative to the wheel carrier body around the x,z-axis with rotation sequence ZX

<pre>.StrRodR.rv_zx.x
<pre>.StrRodR.rv_zx.z
	rad/s	Rotation velocity of the right steering rod relative to the wheel carrier body around the x,z-axis with rotation sequence ZX

<pre>.StrRodR.r_zx.x
<pre>.StrRodR.r_zx.z
	rad	Rotation of the right steering rod relative to the wheel carrier body around the x,z-axis with rotation sequence ZX

<pre>.Vhcl.Bush2DampL.*		Parameters of the strut bushing between the left damper body and vehicle chassis
<pre>.Vhcl.Bush2DampR.*		Parameters of the strut bushing between the right damper body and vehicle chassis

<pre>.WCL.ra_zxy.x
<pre>.WCL.ra_zxy.y
<pre>.WCL.ra_zxy.z
	rad/s^2	Rotation acceleration of the left wheel carrier body relative to vehicle chassis around the x,y,z-axis with rotation sequence ZXY

<pre>.WCL.rv_zxy.x
<pre>.WCL.rv_zxy.y
<pre>.WCL.rv_zxy.z
	rad/s	Rotation velocity of the left wheel carrier body relative to vehicle chassis around the x,y,z-axis with rotation sequence ZXY

Name UAQ	Unit	Info (data type, if not double)

<pre>.WCL.r_zxy.x
<pre>.WCL.r_zxy.y
<pre>.WCL.r_zxy.z
	rad	Rotation of the left wheel carrier body relative to vehicle chassis around the x,y,z-axis with rotation sequence ZXY

<pre>.WCL.tx
<pre>.WCL.ty
<pre>.WCL.tz
	m	Translation of the left wheel carrier body relative to vehicle chassis along the x,y,z-axis

<pre>.WCL.vx
<pre>.WCL.vy
<pre>.WCL.vz
	m/s	Translation velocity of the left wheel carrier body relative to vehicle chassis along the x,y,z-axis

<pre>.WCL.ax
<pre>.WCL.ay
<pre>.WCL.az
	m/s^2	Translation acceleration of the left wheel carrier body relative to vehicle chassis along the x,y,z-axis

<pre>.WCR.ra_zxy.x
<pre>.WCR.ra_zxy.y
<pre>.WCR.ra_zxy.z
	rad/s^2	Rotation acceleration of the right wheel carrier body relative to vehicle chassis around the x,y,z-axis with rotation sequence ZXY

<pre>.WCR.rv_zxy.x
<pre>.WCR.rv_zxy.y
<pre>.WCR.rv_zxy.z
	rad/s	Rotation velocity of the right wheel carrier body relative to vehicle chassis around the x,y,z-axis with rotation sequence ZXY

<pre>.WCR.r_zxy.x
<pre>.WCR.r_zxy.y
<pre>.WCR.r_zxy.z
	rad	Rotation of the right wheel carrier body relative to vehicle chassis around the x,y,z-axis with rotation sequence ZXY

<pre>.WCR.tx
<pre>.WCR.ty
<pre>.WCR.tz
	m	Translation of the right wheel carrier body relative to vehicle chassis along the x,y,z-axis

<pre>.WCR.vx
<pre>.WCR.vy
<pre>.WCR.vz
	m/s	Translation velocity of the right wheel carrier body relative to vehicle chassis along the x,y,z-axis

<pre>.WCR.ax
<pre>.WCR.ay
<pre>.WCR.az
	m/s^2	Translation acceleration of the right wheel carrier body relative to vehicle chassis along the x,y,z-axis

<pre>.WshbL.BushF2Vhcl.*		Parameters of the front bushing between the left wishbone body and vehicle chassis
<pre>.WshbL.BushR2Vhcl.*		Parameters of the rear bushing between the left wishbone body and vehicle chassis

<pre>.WshbL.ra_zyx.x
<pre>.WshbL.ra_zyx.y
<pre>.WshbL.ra_zyx.z
	rad/s^2	Rotation acceleration of the left wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZYX

<pre>.WshbL.rv_zyx.x
<pre>.WshbL.rv_zyx.y
<pre>.WshbL.rv_zyx.z
	rad/s	Rotation velocity of the left wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZYX

<pre>.WshbL.r_zyx.x
<pre>.WshbL.r_zyx.y
<pre>.WshbL.r_zyx.z
	rad	Rotation of the left wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZYX

<pre>.WshbR.BushF2Vhcl.*		Parameters of the front bushing between the right wishbone body and vehicle chassis
<pre>.WshbR.BushR2Vhcl.*		Parameters of the rear bushing between the right wishbone body and vehicle chassis

<pre>.WshbR.ra_zyx.x
<pre>.WshbR.ra_zyx.y
<pre>.WshbR.ra_zyx.z
	rad/s^2	Rotation acceleration of the right wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZYX

<pre>.WshbR.rv_zyx.x
<pre>.WshbR.rv_zyx.y
<pre>.WshbR.rv_zyx.z
	rad/s	Rotation velocity of the right wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZYX

<pre>.WshbR.r_zyx.x
<pre>.WshbR.r_zyx.y
<pre>.WshbR.r_zyx.z
	rad	Rotation of the right wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZYX

24.8.3 McPherson extended suspension

At this point only the additional quantities for the two bodies subframe and steering box are listed. For the description of the other bodies please refer to chapter 24.8.2.'

Name UAQ	Unit	Info (data type, if not double)
<pre> := McPherson.<pos> <pos> := SuspF		

Name UAQ	Unit	Info (data type, if not double)

<pre>.StrBox.ra_zx.x
<pre>.StrBox.ra_zx.z
	rad/s^2	Rotation acceleration of the steering box relative to the subframe body around the x,z-axis with rotation sequence ZX

<pre>.StrBox.rv_zx.x
<pre>.StrBox.rv_zx.z
	rad/s	Rotation velocity of the steering box relative to the subframe body around the x,z-axis with rotation sequence ZX

<pre>.StrBox.r_zx.x
<pre>.StrBox.r_zx.z
	rad	Rotation of the steering box relative to the subframe body around the x,z-axis with rotation sequence ZX

<pre>.StrBox.tx
<pre>.StrBox.ty
<pre>.StrBox.tz
	m	Translation of the steering box relative to the subframe body along the x,y,z-axis

<pre>.StrBox.vx
<pre>.StrBox.vy
<pre>.StrBox.vz
	m/s	Translation velocity of the steering box relative to the subframe body along the x,y,z-axis

<pre>.StrBox.ax
<pre>.StrBox.ay
<pre>.StrBox.az
	m/s^2	Translation acceleration of the steering box relative to the subframe body along the x,y,z-axis

<pre>.Subfrm.BushFL2Vhcl.*		Parameters of the front left bushing between the subframe body and vehicle chassis
<pre>.Subfrm.BushFR2Vhcl.*		Parameters of the front right bushing between the subframe and vehicle chassis
<pre>.Subfrm.BushRL2Vhcl.*		Parameters of the rear left bushing between the subframe body and vehicle chassis
<pre>.Subfrm.BushRR2Vhcl.*		Parameters of the rear right bushing between the subframe and vehicle chassis
<pre>.Subfrm.L2StrBox.*		Parameters of the left bushing between the subframe and the steering box body
<pre>.Subfrm.R2StrBox.*		Parameters of the right bushing between the subframe and the steering box body

<pre>.Subfrm.ra_zyx.x
<pre>.Subfrm.ra_zyx.y
<pre>.Subfrm.ra_zyx.z
	rad/s^2	Rotation acceleration of the subframe body relative to the vehicle chassis around the x,y,z-axis with rotation sequence ZYX

<pre>.Subfrm.rv_zyx.x
<pre>.Subfrm.rv_zyx.y
<pre>.Subfrm.rv_zyx.z
	rad/s	Rotation velocity of the subframe body relative to the vehicle chassis around the x,y,z-axis with rotation sequence ZYX

<pre>.Subfrm.r_zyx.x
<pre>.Subfrm.r_zyx.y
<pre>.Subfrm.r_zyx.z
	rad	Rotation of the subframe body relative to the vehicle chassis around the x,y,z-axis with rotation sequence ZYX

<pre>.Subfrm.tx
<pre>.Subfrm.ty
<pre>.Subfrm.tz
	m	Translation of the subframe body relative to the vehicle chassis along the x,y,z-axis

<pre>.Subfrm.vx
<pre>.Subfrm.vy
<pre>.Subfrm.vz
	m/s	Translation velocity of the subframe body relative to the vehicle chassis along the x,y,z-axis

<pre>.Subfrm.ax
<pre>.Subfrm.ay
<pre>.Subfrm.az
	m/s^2	Translation acceleration of the subframe body relative to the vehicle chassis along the x,y,z-axis

24.8.4 Fourlink suspension

Name UAQ	Unit	Info (data type, if not double)
<pre> := FourLink.<pos> <pos> := SuspR		
<pre>.OSRate	-	Model oversampling rate (integer)
<pre>.PlungL.Frc2Wheel	N	Normal force at the left tire contact point
<pre>.PlungR.Frc2Wheel	N	Normal force at the right tire contact point
<pre>.StbBarL.ra_y	rad/s^2	Rotation acceleration of the left stabilizer bar body relative to vehicle chassis around the y-axis
<pre>.StbBarL.rv_y	rad/s	Rotation velocity of the left stabilizer bar body relative to vehicle chassis around the y-axis
<pre>.StbBarL.r_y	rad	Rotation of the left stabilizer bar body relative to vehicle chassis around the y-axis
<pre>.StbBarR.ra_y	rad/s^2	Rotation acceleration of the right stabilizer bar body relative to vehicle chassis around the y-axis
<pre>.StbBarR.rv_y	rad/s	Rotation velocity of the right stabilizer bar body relative to vehicle chassis around the y-axis
<pre>.StbBarR.r_y	rad	Rotation of the right stabilizer bar body relative to vehicle chassis around the y-axis
<pre>.StbLinkL.Bush.*		Parameters of the bushing between the left stabilizer link and wheel carrier (or lower wishbone) body

<pre>.StbLinkL.ra_yx.x
<pre>.StbLinkL.ra_yx.y
	rad/s^2	Rotation acceleration of the left stabilizer link body relative to stabilizer bar around the x,y-axis with rotation sequence YX

<pre>.StbLinkL.rv_yx.x
<pre>.StbLinkL.rv_yx.y
	rad/s	Rotation velocity of the left stabilizer link body relative to stabilizer bar around the x,y-axis with rotation sequence YX

<pre>.StbLinkL.r_yx.x
<pre>.StbLinkL.r_yx.y
	rad	Rotation of the left stabilizer link body relative to stabilizer bar around the x,y-axis with rotation sequence YX

<pre>.StbLinkR.Bush.*		Parameters of the bushing between the right stabilizer link and wheel carrier (or lower wishbone) body

<pre>.StbLinkR.ra_yx.x
<pre>.StbLinkR.ra_yx.y
	rad/s^2	Rotation acceleration of the right stabilizer link body relative to stabilizer bar around the x,y-axis with rotation sequence YX

<pre>.StbLinkR.rv_yx.x
<pre>.StbLinkR.rv_yx.y
	rad/s	Rotation velocity of the right stabilizer link body relative to stabilizer bar around the x,y-axis with rotation sequence YX

<pre>.StbLinkR.r_yx.x
<pre>.StbLinkRr_yx.y
	rad	Rotation of the right stabilizer link body relative to stabilizer bar around the x,y-axis with rotation sequence YX

<pre>.StrRodL.Bush2Vhcl.*		Parameters of the bushing between the left steering rod body and vehicle chassis

<pre>.StrRodL.ra_zxy.x
<pre>.StrRodL.ra_zxy.y
<pre>.StrRodL.ra_zxy.z
	rad/s^2	Rotation acceleration of the left steering rod relative to the wheel carrier body around the x,y,z-axis with rotation sequence ZXY

<pre>.StrRodL.rv_zxy.x
<pre>.StrRodL.rv_zxy.y
<pre>.StrRodL.rv_zxy.z
	rad/s	Rotation velocity of the left steering rod relative to the wheel carrier body around the x,y,z-axis with rotation sequence ZXY

<pre>.StrRodL.r_zxy.x
<pre>.StrRodL.r_zxy.y
<pre>.StrRodL.r_zxy.z
	rad	Rotation of the left steering rod relative to the wheel carrier body around the x,y,z-axis with rotation sequence ZXY

<pre>.StrRodR.Bush2Vhcl.*		Parameters of the bushing between the right steering rod body and vehicle chassis

<pre>.StrRodR.ra_zxy.x
<pre>.StrRodR.ra_zxy.y
<pre>.StrRodR.ra_zxy.z
	rad/s^2	Rotation acceleration of the right steering rod relative to the wheel carrier body around the x,y,z-axis with rotation sequence ZXY

<pre>.StrRodR.rv_zxy.x
<pre>.StrRodR.rv_zxy.y
<pre>.StrRodR.rv_zxy.z
	rad/s	Rotation velocity of the right steering rod relative to the wheel carrier body around the x,y,z-axis with rotation sequence ZXY

<pre>.StrRodR.r_zxy.x
<pre>.StrRodR.r_zxy.y
<pre>.StrRodR.r_zxy.z
	rad	Rotation of the right steering rod relative to the wheel carrier body around the x,y,z-axis with rotation sequence ZXY

<pre>.TrailArmL.Bush2Vhcl.*		Parameters of the bushing between the left trailing arm body and vehicle chassis
<pre>.TrailArmL.ra_z	rad/s^2	Rotation acceleration of the left trailing arm relative to wheel carrier body around the z-axis
<pre>.TrailArmL.rv_z	rad/s	Rotation velocity of the left trailing arm relative to wheel carrier body around the z-axis
<pre>.TrailArmL.r_z	rad	Rotation of the left trailing arm relative to wheel carrier body around the z-axis
<pre>.TrailArmL.Trq2WC_z	Nm	Spring-damper torque of the left trailing arm body
<pre>.TrailArmR.Bush2Vhcl.*		Parameters of the bushing between the right trailing arm body and vehicle chassis
<pre>.TrailArmR.ra_z	rad/s^2	Rotation acceleration of the right trailing arm relative to wheel carrier body around the z-axis
<pre>.TrailArmR.rv_z	rad/s	Rotation velocity of the right trailing arm relative to wheel carrier body around the z-axis
<pre>.TrailArmR.r_z	rad	Rotation of the right trailing arm relative to wheel carrier body around the z-axis
<pre>.TrailArmR.Trq2WC_z		Spring-damper torque of the l right trailing arm body

<pre>.WCL.ax
<pre>.WCL.ay
<pre>.WCL.az
	m/s^2	Translation acceleration of the left wheel carrier body relative to vehicle chassis along the x,y,z-axis

Name UAQ	Unit	Info (data type, if not double)

<pre>.WCL.ra_zxy.x
<pre>.WCL.ra_zxy.y
<pre>.WCL.ra_zxy.z
	rad/s^2	Rotation acceleration of the left wheel carrier body relative to vehicle chassis around the x,y,z-axis with rotation sequence ZXY

<pre>.WCL.rv_zxy.x
<pre>.WCL.rv_zxy.y
<pre>.WCL.rv_zxy.z
	rad/s	Rotation velocity of the left wheel carrier body relative to vehicle chassis around the x,y,z-axis with rotation sequence ZXY

<pre>.WCL.r_zxy.x
<pre>.WCL.r_zxy.y
<pre>.WCL.r_zxy.z
	rad	Rotation of the left wheel carrier body relative to vehicle chassis around the x,y,z-axis with rotation sequence ZXY

<pre>.WCL.tx
<pre>.WCL.ty
<pre>.WCL.tz
	m	Translation of the left wheel carrier body relative to vehicle chassis along the x,y,z-axis

<pre>.WCL.vx
<pre>.WCL.vy
<pre>.WCL.vz
	m/s	Translation velocity of the left wheel carrier body relative to vehicle chassis along the x,y,z-axis

<pre>.WCR.ax
<pre>.WCR.ay
<pre>.WCR.az
	m/s^2	Translation acceleration of the right wheel carrier body relative to vehicle chassis along the x,y,z-axis

<pre>.WCR.ra_zxy.x
<pre>.WCR.ra_zxy.y
<pre>.WCR.ra_zxy.z
	rad/s^2	Rotation acceleration of the right wheel carrier body relative to vehicle chassis around the x,y,z-axis with rotation sequence ZXY

<pre>.WCR.rv_zxy.x
<pre>.WCR.rv_zxy.y
<pre>.WCR.rv_zxy.z
	rad/s	Rotation velocity of the right wheel carrier body relative to vehicle chassis around the x,y,z-axis with rotation sequence ZXY

<pre>.WCR.r_zxy.x
<pre>.WCR.r_zxy.y
<pre>.WCR.r_zxy.z
	rad	Rotation of the right wheel carrier body relative to vehicle chassis around the x,y,z-axis with rotation sequence ZXY

<pre>.WCR.tx
<pre>.WCR.ty
<pre>.WCR.tz
	m	Translation of the right wheel carrier body relative to vehicle chassis along the x,y,z-axis

<pre>.WCR.vx
<pre>.WCR.vy
<pre>.WCR.vz
	m/s	Translation velocity of the right wheel carrier body relative to vehicle chassis along the x,y,z-axis

<pre>.WishLowL.Bush2Vhcl.*		Parameters of the bushing between the left lower wishbone body and vehicle chassis

<pre>.WishLowL.ra_zxy.x
<pre>.WishLowL.ra_zxy.y
<pre>.WishLowL.ra_zxy.z
	rad/s^2	Rotation acceleration of the left lower wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZXY

<pre>.WishLowL.rv_zxy.x
<pre>.WishLowL.rv_zxy.y
<pre>.WishLowL.rv_zxy.z
	rad/s	Rotation velocity of the left lower wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZXY

<pre>.WishLowL.r_zxy.x
<pre>.WishLowL.r_zxy.y
<pre>.WishLowL.r_zxy.z
	rad	Rotation of the left lower wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZXY

<pre>.WishLowR.Bush2Vhcl.*		Parameters of the bushing between the right lower wishbone body and vehicle chassis

<pre>.WishLowR.ra_zxy.x
<pre>.WishLowR.ra_zxy.y
<pre>.WishLowR.ra_zxy.z
	rad/s^2	Rotation acceleration of the right lower wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZXY

<pre>.WishLowR.rv_zxy.x
<pre>.WishLowR.rv_zxy.y
<pre>.WishLowR.rv_zxy.z
	rad/s	Rotation velocity of the right lower wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZXY

<pre>.WishLowR.r_zxy.x
<pre>.WishLowR.r_zxy.y
<pre>.WishLowR.r_zxy.z
	rad	Rotation of the right lower wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZXY

<pre>.WishUppL.Bush2Vhcl.*		Parameters of the bushing between the left upper wishbone body and vehicle chassis

<pre>.WishUppL.ra_zxy.x
<pre>.WishUppL.ra_zxy.y
<pre>.WishUppL.ra_zxy.z
	rad/s^2	Rotation acceleration of the left upper wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZXY

<pre>.WishUppL.rv_zxy.x
<pre>.WishUppL.rv_zxy.y
<pre>.WishUppL.rv_zxy.z
	rad/s	Rotation velocity of the left upper wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZXY

<pre>.WishUppL.r_zxy.x
<pre>.WishUppL.r_zxy.y
<pre>.WishUppL.r_zxy.z
	rad	Rotation of the left upper wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZXY

<pre>.WishUppR.Bush2Vhcl.*		Parameters of the bushing between the right upper wishbone body and vehicle chassis

<pre>.WishUppR.ra_zxy.x
<pre>.WishUppR.ra_zxy.y
<pre>.WishUppR.ra_zxy.z
	rad/s^2	Rotation acceleration of the right upper wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZXY

Name UAQ	Unit	Info (data type, if not double)

<pre>.WishUppR.rv_zxy.x
<pre>.WishUppR.rv_zxy.y
<pre>.WishUppR.rv_zxy.z
	rad/s	Rotation velocity of the right upper wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZXY

<pre>.WishUppR.r_zxy.x
<pre>.WishUppR.r_zxy.y
<pre>.WishUppR.r_zxy.z
	rad	Rotation of the right upper wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZXY

24.8.5 Fourlink extended suspension

At this point only the additional quantities for the body subframe are listed. For the description of the other bodies please refer to chapter 24.8.4.

Name UAQ	Unit	Info (data type, if not double)
<pre> := FourLink.<pos> <pos> := SuspR		
<pre>.Subfrm.BushFL2Vhcl.*		Parameters of the front left bushing between the subframe body and vehicle chassis
<pre>.Subfrm.BushFR2Vhcl.*		Parameters of the front right bushing between the subframe and vehicle chassis
<pre>.Subfrm.BushRL2Vhcl.*		Parameters of the rear left bushing between the subframe body and vehicle chassis
<pre>.Subfrm.BushRR2Vhcl.*		Parameters of the rear right bushing between the subframe and vehicle chassis

<pre>.Subfrm.ra_zyx.x
<pre>.Subfrm.ra_zyx.y
<pre>.Subfrm.ra_zyx.z
	rad/s^2	Rotation acceleration of the subframe body relative to the vehicle chassis around the x,y,z-axis with rotation sequence ZYX

<pre>.Subfrm.rv_zyx.x
<pre>.Subfrm.rv_zyx.y
<pre>.Subfrm.rv_zyx.z
	rad/s	Rotation velocity of the subframe body relative to the vehicle chassis around the x,y,z-axis with rotation sequence ZYX

<pre>.Subfrm.r_zyx.x
<pre>.Subfrm.r_zyx.y
<pre>.Subfrm.r_zyx.z
	rad	Rotation of the subframe body relative to the vehicle chassis around the x,y,z-axis with rotation sequence ZYX

<pre>.Subfrm.tx
<pre>.Subfrm.ty
<pre>.Subfrm.tz
	m	Translation of the subframe body relative to the vehicle chassis along the x,y,z-axis

<pre>.Subfrm.vx
<pre>.Subfrm.vy
<pre>.Subfrm.vz
	m/s	Translation velocity of the subframe body relative to the vehicle chassis along the x,y,z-axis

<pre>.Subfrm.ax
<pre>.Subfrm.ay
<pre>.Subfrm.az
	m/s^2	Translation acceleration of the subframe body relative to the vehicle chassis along the x,y,z-axis

24.8.6 Twistbeam suspension

Name UAQ	Unit	Info (data type, if not double)
<pre> := TwistBeam.<pos> <pos> := SuspR		
<pre>.OSRate	-	Model oversampling rate (integer)
<pre>.PlungL.Frc2Wheel	N	Normal force at the left tire contact point
<pre>.PlungR.Frc2Wheel	N	Normal force at the right tire contact point

<pre>.TorsBeam.ra_zyx.x
<pre>.TorsBeam.ra_zyx.y
<pre>.TorsBeam.ra_zyx.z
	rad/s^2	Rotation acceleration of the torsion beam body relative to the vehicle chassis around the x,y,z-axis with rotation sequence ZYX

<pre>.TorsBeam.rv_zyx.x
<pre>.TorsBeam.rv_zyx.y
<pre>.TorsBeam.rv_zyx.z
	rad/s	Rotation velocity of the torsion beam body relative to the vehicle chassis around the x,y,z-axis with rotation sequence ZYX

<pre>.TorsBeam.r_zyx.x
<pre>.TorsBeam.r_zyx.y
<pre>.TorsBeam.r_zyx.z
	rad	Rotation of the torsion beam body relative to the vehicle chassis around the x,y,z-axis with rotation sequence ZYX

<pre>.TorsBeam.tx
<pre>.TorsBeam.ty
<pre>.TorsBeam.tz
	m	Translation of the torsion beam body relative to the vehicle chassis along the x,y,z-axis

<pre>.TorsBeam.vx
<pre>.TorsBeam.vy
<pre>.TorsBeam.vz
	m/s	Translation velocity of the torsion beam body relative to the vehicle chassis along the x,y,z-axis

<pre>.TorsBeam.ax
<pre>.TorsBeam.ay
<pre>.TorsBeam.az
	m/s^2	Translation acceleration of the torsion beam body relative to vehicle chassis along the x,y,z-axis

<pre>.TrlArmL.Bush2Vhcl.*		Parameters of the bushing between the left trailing arm body and vehicle chassis

<pre>.TrlArmL.ra_zyx.x
<pre>.TrlArmL.ra_zyx.y
<pre>.TrlArmL.ra_zyx.z
	rad/s^2	Rotation acceleration of the left trailing arm body relative to torsion beam around the x,y,z-axis with rotation sequence ZYX

<pre>.TrlArmL.rv_zyx.x
<pre>.TrlArmL.rv_zyx.y
<pre>.TrlArmL.rv_zyx.z
	rad/s	Rotation velocity of the left trailing arm body relative to torsion beam around the x,y,z-axis with rotation sequence ZYX

<pre>.TrlArmL.r_zyx.x
<pre>.TrlArmL.r_zyx.y
<pre>.TrlArmL.r_zyx.z
	rad	Rotation of the left trailing arm body relative to torsion beam around the x,y,z-axis with rotation sequence ZYX

<pre>.TrlArmL.Trq2TorsBeam.x
<pre>.TrlArmL.Trq2TorsBeam.y
<pre>.TrlArmL.Trq2TorsBeam.z
	Nm	Spring-damper torque of the left trailing arm body to the torsion beam around the x,y,z-axis

<pre>.TrlArmR.Bush2Vhcl.*		Parameters of the bushing between the right trailing arm body and vehicle chassis

<pre>.TrlArmR.ra_zyx.x
<pre>.TrlArmR.ra_zyx.y
<pre>.TrlArmR.ra_zyx.z
	rad/s^2	Rotation acceleration of the right trailing arm body relative to torsion beam around the x,y,z-axis with rotation sequence ZYX

<pre>.TrlArmR.rv_zyx.x
<pre>.TrlArmR.rv_zyx.y
<pre>.TrlArmR.rv_zyx.z
	rad/s	Rotation velocity of the right trailing arm body relative to torsion beam around the x,y,z-axis with rotation sequence ZYX

<pre>.TrlArmR.r_zyx.x
<pre>.TrlArmR.r_zyx.y
<pre>.TrlArmR.r_zyx.z
	rad	Rotation of the right trailing arm body relative to torsion beam around the x,y,z-axis with rotation sequence ZYX

<pre>.TrlArmR.Trq2TorsBeam.x
<pre>.TrlArmR.Trq2TorsBeam.y
<pre>.TrlArmR.Trq2TorsBeam.z
	Nm	Spring-damper torque of the right trailing arm body to the torsion beam around the x,y,z-axis

<pre>.WCL.ra_z	rad/s^2	Rotation acceleration of the left wheel carrier body relative to trailing body arm around the z-axis
<pre>.WCL.rv_z	rad/s	Rotation velocity of the left wheel carrier body relative to trailing arm body around the z-axis
<pre>.WCL.r_z	rad	Rotation of the left wheel carrier body relative to trailing arm body around the z-axis
<pre>.WCL.Trq2TrailArm_z	Nm	Spring-damper torque of the left wheel carrier body to trailing arm body
<pre>.WCR.ra_z	rad/s^2	Rotation acceleration of the right wheel carrier body relative to trailing body arm around the z-axis
<pre>.WCR.rv_z	rad/s	Rotation velocity of the right wheel carrier body relative to trailing arm body around the z-axis
<pre>.WCR.r_z	rad	Rotation of the right wheel carrier body relative to trailing arm body around the z-axis
<pre>.WCR.Trq2TrailArm_z	Nm	Spring-damper torque of the right wheel carrier body to trailing arm body

24.8.7 Twistbeam extended suspension

At this point only the quantities for the additional model elements are listed. For the description of the other bodies please refer to chapter 24.8.6.'

Name UAQ	Unit	Info (data type, if not double)
<pre> := TwistBeam.<pos> <pos> := SuspR		
<pre>.TrlArmL.Bush2TorsBeam.*		Parameters of the bushing between the left trailing arm body and the torsion beam body

<pre>.TrlArmL.tx
<pre>.TrlArmL.ty
<pre>.TrlArmL.tz
	m	Translation of the left trailing arm body relative to the torsion beam along the x,y,z-axis

<pre>.TrlArmL.vx
<pre>.TrlArmL.vy
<pre>.TrlArmL.vz
	m/s	Translation velocity of the left trailing arm body relative to the torsion beam along the x,y,z-axis

<pre>.TrlArmL.ax
<pre>.TrlArmL.ay
<pre>.TrlArmL.az
	m/s^2	Translation acceleration of the left trailing arm body relative to the torsion beam along the x,y,z-axis

<pre>.TrlArmR.Bush2TorsBeam.*		Parameters of the bushing between the rigth trailing arm body and the torsion beam body

<pre>.TrlArmR.tx
<pre>.TrlArmR.ty
<pre>.TrlArmR.tz
	m	Translation of the rigth trailing arm body relative to the torsion beam along the x,y,z-axis

<pre>.TrlArmR.vx
<pre>.TrlArmR.vy
<pre>.TrlArmR.vz
	m/s	Translation velocity of the rigth trailing arm body relative to the torsion beam along the x,y,z-axis

<pre>.TrlArmR.ax
<pre>.TrlArmR.ay
<pre>.TrlArmR.az
	m/s^2	Translation acceleration of the rigth trailing arm body relative to the torsion beam along the x,y,z-axis

24.8.8 Double wishbone suspension

Name UAQ	Unit	Info (data type, if not double)
<pre> := DoubleWishbone.<pos> <pos> := SuspF		
<pre>.OSRate	-	Model oversampling rate (integer)
<pre>.PlungL.Frc2Wheel	N	Normal force at the left tire contact point
<pre>.PlungR.Frc2Wheel	N	Normal force at the right tire contact point
<pre>.StbBarL.ra_y	rad/s^2	Rotation acceleration of the left stabilizer bar body relative to vehicle chassis around the y-axis
<pre>.StbBarL.rv_y	rad/s	Rotation velocity of the left stabilizer bar body relative to vehicle chassis around the y-axis
<pre>.StbBarL.r_y	rad	Rotation of the left stabilizer bar body relative to vehicle chassis around the y-axis
<pre>.StbBarR.ra_y	rad/s^2	Rotation acceleration of the right stabilizer bar body relative to vehicle chassis around the y-axis
<pre>.StbBarR.rv_y	rad/s	Rotation velocity of the right stabilizer bar body relative to vehicle chassis around the y-axis
<pre>.StbBarR.r_y	rad	Rotation of the right stabilizer bar body relative to vehicle chassis around the y-axis
<pre>.StbLinkL.Bush2WC.*		Parameters of the bushing between the left stabilizer link and wheel carrier body
<pre>.StbLinkR.Bush2WC.*		Parameters of the bushing between the right stabilizer link and wheel carrier body

<pre>.StbLinkL.ra_yx.x
<pre>.StbLinkL.ra_yx.y
	rad/s^2	Rotation acceleration of the left stabilizer link body relative to stabilizer bar around the x,y-axis with rotation sequence YX

<pre>.StbLinkL.rv_yx.x
<pre>.StbLinkL.rv_yx.y
	rad/s	Rotation velocity of the left stabilizer link body relative to stabilizer bar around the x,y-axis with rotation sequence YX

<pre>.StbLinkL.r_yx.x
<pre>.StbLinkL.r_yx.y
	rad	Rotation of the left stabilizer link body relative to stabilizer bar around the x,y-axis with rotation sequence YX

<pre>.StbLinkR.ra_yx.x
<pre>.StbLinkR.ra_yx.y
	rad/s^2	Rotation acceleration of the right stabilizer link body relative to stabilizer bar around the x,y-axis with rotation sequence YX

<pre>.StbLinkR.rv_yx.x
<pre>.StbLinkR.rv_yx.y
	rad/s	Rotation velocity of the right stabilizer link body relative to stabilizer bar around the x,y-axis with rotation sequence YX

<pre>.StbLinkR.r_yx.x
<pre>.StbLinkR.r_yx.y
	rad	Rotation of the right stabilizer link body relative to stabilizer bar around the x,y-axis with rotation sequence YX

<pre>.StrRack.a_y	m/s^2	Translation acceleration of the steering rack body relative to the vehicle chassis along the y-axis (information imposed by steering model)
<pre>.StrRack.t_y	m	Translation of the steering rack body relative to the vehicle chassis along the y-axis (information imposed by steering model)
<pre>.StrRack.v_y	m/s	Translation velocity of the steering rack body relative to the vehicle chassis along the y-axis (information imposed by steering model)
<pre>.StrRodL.Bush2StrRack.*		Parameters of the bushing between the left steering rod and steering rack body
<pre>.StrRodR.Bush2StrRack.*		Parameters of the bushing between the right steering rod and steering rack body

<pre>.StrRodL.ra_zx.x
<pre>.StrRodL.ra_zx.z
	rad/s^2	Rotation acceleration of the left steering rod relative to the wheel carrier body around the x,z-axis with rotation sequence ZX

<pre>.StrRodL.rv_zx.x
<pre>.StrRodL.rv_zx.z
	rad/s	Rotation velocity of the left steering rod relative to the wheel carrier body around the x,z-axis with rotation sequence ZX

<pre>.StrRodL.r_zx.x
<pre>.StrRodL.r_zx.z
	rad	Rotation of the left steering rod relative to the wheel carrier body around the x,z-axis with rotation sequence ZX

<pre>.StrRodR.ra_zx.x
<pre>.StrRodR.ra_zx.z
	rad/s^2	Rotation acceleration of the right steering rod relative to the wheel carrier body around the x,z-axis with rotation sequence ZX

<pre>.StrRodR.rv_zx.x
<pre>.StrRodR.rv_zx.z
	rad/s	Rotation velocity of the right steering rod relative to the wheel carrier body around the x,z-axis with rotation sequence ZX

<pre>.StrRodR.r_zx.x
<pre>.StrRodR.r_zx.z
	rad	Rotation of the right steering rod relative to the wheel carrier body around the x,z-axis with rotation sequence ZX

<pre>.Vhcl.Bush2DampL.*		Parameters of the strut bushing between the left damper body and vehicle chassis
<pre>.Vhcl.Bush2DampR.*		Parameters of the strut bushing between the right damper body and vehicle chassis

<pre>.WCL.ax
<pre>.WCL.ay
<pre>.WCL.az
	m/s^2	Translation acceleration of the left wheel carrier body relative to vehicle chassis along the x,y,z-axis

<pre>.WCL.ra_zxy.x
<pre>.WCL.ra_zxy.y
<pre>.WCL.ra_zxy.z
	rad/s^2	Rotation acceleration of the left wheel carrier body relative to vehicle chassis around the x,y,z-axis with rotation sequence ZXY

<pre>.WCL.rv_zxy.x
<pre>.WCL.rv_zxy.y
<pre>.WCL.rv_zxy.z
	rad/s	Rotation velocity of the left wheel carrier body relative to vehicle chassis around the x,y,z-axis with rotation sequence ZXY

<pre>.WCL.r_zxy.x
<pre>.WCL.r_zxy.y
<pre>.WCL.r_zxy.z
	rad	Rotation of the left wheel carrier body relative to vehicle chassis around the x,y,z-axis with rotation sequence ZXY

Name UAQ	Unit	Info (data type, if not double)

<pre>.WCL.tx
<pre>.WCL.ty
<pre>.WCL.tz
	m	Translation of the left wheel carrier body relative to vehicle chassis along the x,y,z-axis

<pre>.WCL.vx
<pre>.WCL.vy
<pre>.WCL.vz
	m/s	Translation velocity of the left wheel carrier body relative to vehicle chassis along the x,y,z-axis

<pre>.WCR.ax
<pre>.WCR.ay
<pre>.WCR.az
	m/s^2	Translation acceleration of the right wheel carrier body relative to vehicle chassis along the x,y,z-axis

<pre>.WCR.ra_zxy.x
<pre>.WCR.ra_zxy.y
<pre>.WCR.ra_zxy.z
	rad/s^2	Rotation acceleration of the right wheel carrier body relative to vehicle chassis around the x,y,z-axis with rotation sequence ZXY

<pre>.WCR.rv_zxy.x
<pre>.WCR.rv_zxy.y
<pre>.WCR.rv_zxy.z
	rad/s	Rotation velocity of the right wheel carrier body relative to vehicle chassis around the x,y,z-axis with rotation sequence ZXY

<pre>.WCR.r_zxy.x
<pre>.WCR.r_zxy.y
<pre>.WCR.r_zxy.z
	rad	Rotation of the right wheel carrier body relative to vehicle chassis around the x,y,z-axis with rotation sequence ZXY

<pre>.WCR.tx
<pre>.WCR.ty
<pre>.WCR.tz
	m	Translation of the right wheel carrier body relative to vehicle chassis along the x,y,z-axis

<pre>.WCR.vx
<pre>.WCR.vy
<pre>.WCR.vz
	m/s	Translation velocity of the right wheel carrier body relative to vehicle chassis along the x,y,z-axis

<pre>.WishLowL.BushF2Vhcl.*		Parameters of the front bushing between the left lower wishbone body and vehicle chassis
<pre>.WishLowL.BushR2Vhcl.*		Parameters of the rear bushing between the left lower wishbone body and vehicle chassis

<pre>.WishLowL.ra_zyx.x
<pre>.WishLowL.ra_zyx.y
<pre>.WishLowL.ra_zyx.z
	rad/s^2	Rotation acceleration of the left lower wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZYX

<pre>.WishLowL.rv_zyx.x
<pre>.WishLowL.rv_zyx.y
<pre>.WishLowL.rv_zyx.z
	rad/s	Rotation velocity of the left lower wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZYX

<pre>.WishLowL.r_zyx.x
<pre>.WishLowL.r_zyx.y
<pre>.WishLowL.r_zyx.z
	rad	Rotation of the left lower wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZYX

<pre>.WishLowR.BushF2Vhcl.*		Parameters of the front bushing between the right lower wishbone body and vehicle chassis
<pre>.WishLowR.BushR2Vhcl.*		Parameters of the rear bushing between the right lower wishbone body and vehicle chassis

<pre>.WishLowR.ra_zyx.x
<pre>.WishLowR.ra_zyx.y
<pre>.WishLowR.ra_zyx.z
	rad/s^2	Rotation acceleration of the right lower wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZYX

<pre>.WishLowR.rv_zyx.x
<pre>.WishLowR.rv_zyx.y
<pre>.WishLowR.rv_zyx.z
	rad/s	Rotation velocity of the right lower wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZYX

<pre>.WishLowR.r_zyx.x
<pre>.WishLowR.r_zyx.y
<pre>.WishLowR.r_zyx.z
	rad	Rotation of the right lower wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZYX

<pre>.WishUppL.BushF2Vhcl.*		Parameters of the front bushing between the left upper wishbone body and vehicle chassis
<pre>.WishUppL.BushR2Vhcl.*		Parameters of the rear bushing between the left upper wishbone body and vehicle chassis

<pre>.WishUppL.ra_zyx.x
<pre>.WishUppL.ra_zyx.y
<pre>.WishUppL.ra_zyx.z
	rad/s^2	Rotation acceleration of the left upper wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZYX

<pre>.WishUppL.rv_zyx.x
<pre>.WishUppL.rv_zyx.y
<pre>.WishUppL.rv_zyx.z
	rad/s	Rotation velocity of the left upper wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZYX

<pre>.WishUppL.r_zyx.x
<pre>.WishUppL.r_zyx.y
<pre>.WishUppL.r_zyx.z
	rad	Rotation of the left upper wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZYX

<pre>.WishUppR.BushF2Vhcl.*		Parameters of the front bushing between the right upper wishbone body and vehicle chassis
<pre>.WishUppR.BushR2Vhcl.*		Parameters of the rear bushing between the right upper wishbone body and vehicle chassis

<pre>.WishUppR.ra_zyx.x
<pre>.WishUppR.ra_zyx.y
<pre>.WishUppR.ra_zyx.z
	rad/s^2	Rotation acceleration of the right upper wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZYX

<pre>.WishUppR.rv_zyx.x
<pre>.WishUppR.rv_zyx.y
<pre>.WishUppR.rv_zyx.z
	rad/s	Rotation velocity of the right upper wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZYX

Name UAQ	Unit	Info (data type, if not double)

<pre>.WishUppR.r_zyx.x
<pre>.WishUppR.r_zyx.y
<pre>.WishUppR.r_zyx.z
	rad	Rotation of the right upper wishbone relative to wheel carrier body around the x,y,z-axis with rotation sequence ZYX

24.8.9 Double wishbone extended suspension

At this point only the additional quantities for the two bodies subframe and steering box are listed. For the description of the other bodies please refer to chapter 24.8.8.

Name UAQ	Unit	Info (data type, if not double)
<pre> := DoubleWishbone.<pos> <pos> := SuspF		

<pre>.StrBox.ra_zx.x
<pre>.StrBox.ra_zx.z
	rad/s^2	Rotation acceleration of the steering box relative to the subframe body around the x,z-axis with rotation sequence ZX

<pre>.StrBox.rv_zx.x
<pre>.StrBox.rv_zx.z
	rad/s	Rotation velocity of the steering box relative to the subframe body around the x,z-axis with rotation sequence ZX

<pre>.StrBox.r_zx.x
<pre>.StrBox.r_zx.z
	rad	Rotation of the steering box relative to the subframe body around the x,z-axis with rotation sequence ZX

<pre>.StrBox.tx
<pre>.StrBox.ty
<pre>.StrBox.tz
	m	Translation of the steering box relative to the subframe body along the x,y,z-axis

<pre>.StrBox.vx
<pre>.StrBox.vy
<pre>.StrBox.vz
	m/s	Translation velocity of the steering box relative to the subframe body along the x,y,z-axis

<pre>.StrBox.ax
<pre>.StrBox.ay
<pre>.StrBox.az
	m/s^2	Translation acceleration of the steering box relative to the subframe body along the x,y,z-axis

<pre>.Subfrm.BushFL2Vhcl.*		Parameters of the front left bushing between the subframe body and vehicle chassis
<pre>.Subfrm.BushFR2Vhcl.*		Parameters of the front right bushing between the subframe and vehicle chassis
<pre>.Subfrm.BushRL2Vhcl.*		Parameters of the rear left bushing between the subframe body and vehicle chassis
<pre>.Subfrm.BushRR2Vhcl.*		Parameters of the rear right bushing between the subframe and vehicle chassis
<pre>.Subfrm.L2StrBox.*		Parameters of the left bushing between the subframe and the steering box body
<pre>.Subfrm.R2StrBox.*		Parameters of the right bushing between the subframe and the steering box body

<pre>.Subfrm.ra_zyx.x
<pre>.Subfrm.ra_zyx.y
<pre>.Subfrm.ra_zyx.z
	rad/s^2	Rotation acceleration of the subframe body relative to the vehicle chassis around the x,y,z-axis with rotation sequence ZYX

<pre>.Subfrm.rv_zyx.x
<pre>.Subfrm.rv_zyx.y
<pre>.Subfrm.rv_zyx.z
	rad/s	Rotation velocity of the subframe body relative to the vehicle chassis around the x,y,z-axis with rotation sequence ZYX

<pre>.Subfrm.r_zyx.x
<pre>.Subfrm.r_zyx.y
<pre>.Subfrm.r_zyx.z
	rad	Rotation of the subframe body relative to the vehicle chassis around the x,y,z-axis with rotation sequence ZYX

<pre>.Subfrm.tx
<pre>.Subfrm.ty
<pre>.Subfrm.tz
	m	Translation of the subframe body relative to the vehicle chassis along the x,y,z-axis

<pre>.Subfrm.vx
<pre>.Subfrm.vy
<pre>.Subfrm.vz
	m/s	Translation velocity of the subframe body relative to the vehicle chassis along the x,y,z-axis

<pre>.Subfrm.ax
<pre>.Subfrm.ay
<pre>.Subfrm.az
	m/s^2	Translation acceleration of the subframe body relative to the vehicle chassis along the x,y,z-axis

24.9 Steering System

24.9.1 Steering System -General

Name UAQ	Name C-Code	Name CM4SL	Unit	Info (data type, if not double)
<pre> := Steering.IF				
Steer.AssistFrc	<pre>.AssistFrc	Steering IF AssistFrc	N	Assisting force at steering rack
Steer.AssistFrc_Ext	<pre>.AssistFrc_Ext	-	N	Assisting external force at steering rack
Steer.AssistTrqCol	<pre>.AssistTrqCol	Steering IF AssistTrqCol	Nm	Assisting torque at column
Steer.AssistTrqCol_Ext	<pre>.AssistTrqCol_Ext	-	Nm	Assisting external torque at column
Steer.AssistTrqPin	<pre>.AssistTrqPin	Steering IF AssistTrqPin	Nm	Assisting torque at pinion
Steer.AssistTrqPin_Ext	<pre>.AssistTrqPin_Ext	-	Nm	Assisting external torque at pinion
Steer.Cfg.PosSign	<pre>.CfgIF->PosSign	-	-	Sign of the steering rack movement, depending on the front suspension kinematics: +1: positive rack movement leads to a positive wheel rotation around z-axis -1: positive rack movement leads to a negative wheel rotation around z-axis

Steer.L.iSteer2q
Steer.R.iSteer2q
	<pre>.L.iSteer2q
	<pre>.R.iSteer2q
		Steering IF L iSteer2q
		Steering IF R iSteer2q
			m/rad	Current ratio steering wheel angle to generalized steering coordinate on left/right side

Steer.L.q
Steer.R.q
	<pre>.L.q
	<pre>.R.q
		Steering IF L q
		Steering IF R q
			m	Generalized steering coordinate for the front left/right suspension

Steer.L.qp
Steer.R.qp
	<pre>.L.qp
	<pre>.R.qp
		Steering IF L qp
		Steering IF R qp
			m/s	Generalized steering velocity for the front left/right suspension

Steer.L.qpp
Steer.R.qpp
	<pre>.L.qpp
	<pre>.R.qpp
		Steering IF L qpp
		Steering IF R qpp
			m/s^2	Generalized steering acceleration for the front left/right suspension

Steer.RL.q Steer.RR.q	Steering.RL.q Steering.RR.q		m	Generalized steering coordinate for the rear left/right suspension
Steer.RL.qp Steer.RR.qp	Steering.RL.qp Steering.RR.qp		m/s	Generalized steering velocity for the rear left/right suspension
Steer.SteerBy	<pre>.SteerBy	-	-	Type of steering mode: ([0..2]) 1 = Steer by Angle, 2 = Steer by Torque.
Steer.WhlAcc	<pre>.AngAcc	Steering IF AngAcc	rad/s^2	Steering wheel rotational acceleration.
Steer.WhlAng	<pre>.Ang	Steering IF Ang	rad	Steering wheel angle
Steer.WhlTrq	<pre>.Trq	Steering IF Trq	Nm	Steering wheel torque (input torque from DrivMan/VehicleControl or internal feedback torque)
Steer.WhlTrqStatic	<pre>.TrqStatic	Steering IF TrqStatic	Nm	Steering wheel torque required to compensate wheel forces to steering rack (static conditions)
Steer.WhlVel	<pre>.AngVel	Steering IF AngVel	rad/s	Steering wheel rotational speed

24.9.2 Steering System - Pfeffer

Name UAQ	Unit	Info (data type, if not double)
<pre> := Steer.Pfeffer		
<pre>.EPS.iA	A	Internal motor current
<pre>.EPS.iA_stat	A	Target motor current (from boost curve)

<pre>.EPS.Motor.q
<pre>.EPS.Motor.qp
<pre>.EPS.Motor.qpp
	rad
	rad/s
	rad/s^2
		Degree of freedom of the motor: angle,
		rotational velocity, rotational acceleration

<pre>.EPS.RCBN.Frc_Fric	N	Exponential spring friction force in the recirculating ball nut system

<pre>.EPS.RCBN.qBN
<pre>.EPS.RCBN.qpBN
<pre>.EPS.RCBN.qppBN
	m
	m/s
	m/s^2
		First degree of freedom of recirculating ball nut system: position, velocity, acceleration

<pre>.EPS.RCBN.qKGT
<pre>.EPS.RCBN.qpKGT
<pre>.EPS.RCBN.qppKGT
	rad
	rad/s
	rad/s^2
		Second degree of freedom of recirculating ball nut system: angle, rotational velocity, rotational acceleration

Name UAQ	Unit	Info (data type, if not double)
<pre>.EPS.RCBN.Trq_Fric	Nm	Exponential spring friction torque in the recirculating ball nut system
<pre>.EPS.Trq_Demand	Nm	Demanded torque for the PI-Controller
<pre>.EPS.Trq_E	Nm	Motor torque (iA*Kt)
<pre>.EPS.Trq_Eext	Nm	External motor torque
<pre>.EPS.Trq_L	Nm	Torque in the spring-damper element of the belt
<pre>.HPS.pDelta	Pa	Internal pressure difference between the chambers
<pre>.HPS.pDelta_stat	Pa	Target pressure difference (from boost curve)

<pre>.HPS.pL
<pre>.HPS.pR
	Pa	Internal pressure in the left/right chamber

<pre>.HPS.pL_stat
<pre>.HPS.pR_stat
	Pa	Target pressure in the left/right chamber

<pre>.HPS.Trq_Pump	Nm	Required pump torque

<pre>.HPS.VolL
<pre>.HPS.VolR
	m^3	Volume in the left/right chamber

<pre>.Mech.Column.iUpp2Low	-	Non-uniformity ratio from upper to lower column
<pre>.Mech.Column.LowAng	rad	Lower column angle
<pre>.Mech.Column.Trq_Damp	Nm	Damping torque in the upper column
<pre>.Mech.Column.Trq_Fric	Nm	Exponential spring friction torque in the column
<pre>.Mech.Column.UppAng	rad	Upper column angle
<pre>.Mech.DeltaAng	rad	Relative angle between steering wheel angle and the pinion angle, contains the torsion of all steering shaft components (column, hardy disc, torsion bar)
<pre>.Mech.DeltaAngVel	rad/s	Derivation of the relative angle between steering wheel angle and the pinion angle
<pre>.Mech.Rack.Frc_Damp	N	Damping force in the rack
<pre>.Mech.Rack.Frc_Fric	N	Exponential spring friction force in the rack
<pre>.Mech.Torsbar.Trq_Fric	Nm	Exponential spring friction torque in the torsion bar
<pre>.Mech.Torsbar.Trq_Twist	Nm	Twist torque in the torsion bar
<pre>.Mech.Torsbar.TwistAng	rad	Twist angle in the torsion bar
<pre>.Mech.Trq_LowSprDamp	Nm	Torque in the lower spring-damper element
<pre>.Mech.Trq_Pinion	Nm	Torque in the pinion
<pre>.Mech.Trq_UppSprDamp	Nm	Torque in the upper spring-damper element

24.10 Tire

24.10.1 Tire Model Tame Tire

Name UAQ	Unit	Info (data type, if not integer)
<pos> := FL, FR, RL, RR or FL.Twin, FR.Twin, RL.Twin, RR.Twin		
TameTire.<pos>.Press	bar	Inflation pressure at <pos>
TameTire.<pos>.TempCore	degC	Core temperature of the compound at <pos>
TameTire.<pos>.TempCoreAvg	degC	Average core temperature of the compound at <pos>
TameTire.<pos>.TempSurf	degC	Surface temperature of the compound at <pos>
TameTire.<pos>.TempTrack	degC	Track temperature at <pos>

24.10.2 Tire Model Magic Formula 5.2 and 6.1

Name UAQ	Unit	Info (data type, if not integer)
MFTire.<pos>.Dx	N	Peak value of pure longitudinal force at <pos>
MFTire.<pos>.Dy	N	Peak value of pure lateral force at <pos>
MFTire.<pos>.Press	bar	Tire inflation pressure at <pos> (MF 6.1 only)
<pos> := FL, FR, RL, RR or FL.Twin, FR.Twin, RL.Twin, RR.Twin		

24.10.3 Tire Model MF-Tyre/MF-Swift 6.2

Name UAQ	Unit	Info (data type, if not integer)
<pre> := TNO_MFS.<pos> <pos> := FL, FR, RL, RR or FL.Twin, FR.Twin, RL.Twin, RR.Twin		
<pre>.idtyre	-	Tire ID used for the MF-Tyre/MF-Swift library
<pre>.contact_point_force_longitudinal	N	longitudinal force
<pre>.contact_point_force_lateral	N	lateral force
<pre>.contact_point_force_vertical	N	vertical force
<pre>.contact_point_moment_roll	Nm	overturning force
<pre>.contact_point_moment_pitch	Nm	rolling resistance moment
<pre>.contact_point_moment_yaw	Nm	self-aligning moment
<pre>.slip_ratio_longitudinal	-	longitudinal slip
<pre>.slip_angle_lateral	rad	side slip angle
<pre>.inclination_angle	rad	inclination angle
<pre>.turn_slip	1/m	turn slip
<pre>.contact_point_velocity_longitudinal	m/s	wheel contact centre forward velocity
<pre>.loaded_radius	m	loaded radius
<pre>.effective_rolling_radius	m	effective rolling radius
<pre>.deflection_vertical	m	tyre deflection
<pre>.contact_patch_length	m	tyre contact length
<pre>.pneumatic_trail	m	pneimatic trail
<pre>.peak_friction_longitudinal	-	longitudinal friction coefficient
<pre>.peak_friction_lateral	-	llateral friction coefficient
<pre>.relaxation_length_longitudinal	m	longitudinal relaxation length
<pre>.relaxation_length_lateral	m	lateral relaxation length
<pre>.slip_velocity_longitudinal	m/s	longitudinal wheel slip velocity
<pre>.slip_velocity_lateral	m/s	lateral wheel slip velocity
<pre>.compression_velocity_vertical	m/s	tyre compression velocity
<pre>.angular_velocity_yaw	m/s	tyre yaw velocity
<pre>.contact_point_coordinate_x	m	global x coordinate contact point
<pre>.contact_point_coordinate_y	m	global y coordinate contact point
<pre>.contact_point_coordinate_z	m	global z coordinate contact point
<pre>.road_normal_component_x	-	global x component road normal
<pre>.road_normal_component_y	-	global y component contact point
<pre>.road_normal_component_z	-	global z component contact point
<pre>.effective_road_height	m	effective road height
<pre>.effective_road_slope	rad	effective forward slope
<pre>.effective_road_curvature	1/m	effective road curvature
<pre>.effective_road_banking	rad	effective road banking/road camber angle

24.10.4 Tire Model MF-Tyre/MF-Swift 7.2

Name UAQ	Unit	Info (data type, if not integer)
<pre> := TASS_MFS.<pos>		
<pos> := FL, FR, RL, RR or FL.Twin, FR.Twin, RL.Twin, RR.Twin		
<pre>.idtyre	-	Tire ID used for the MF-Tyre/MF-Swift library
<pre>.contact_point_force_longitudinal	N	longitudinal force
<pre>.contact_point_force_lateral	N	lateral force
<pre>.contact_point_force_vertical	N	vertical force
<pre>.contact_point_moment_roll	Nm	overturning force
<pre>.contact_point_moment_pitch	Nm	rolling resistance moment
<pre>.contact_point_moment_yaw	Nm	self-aligning moment
<pre>.slip_ratio_longitudinal	-	longitudinal slip
<pre>.slip_angle_lateral	rad	side slip angle
<pre>.inclination_angle	rad	inclination angle
<pre>.turn_slip	1/m	turn slip
<pre>.contact_point_velocity_longitudinal	m/s	wheel contact centre forward velocity
<pre>.loaded_radius	m	loaded radius
<pre>.effective_rolling_radius	m	effective rolling radius
<pre>.deflection_vertical	m	tyre deflection
<pre>.contact_patch_length	m	tyre contact length
<pre>.pneumatic_trail	m	pneimatic trail
<pre>.peak_friction_longitudinal	-	longitudinal friction coefficient
<pre>.peak_friction_lateral	-	llateral friction coefficient
<pre>.relaxation_length_longitudinal	m	longitudinal relaxation length
<pre>.relaxation_length_lateral	m	lateral relaxation length
<pre>.slip_velocity_longitudinal	m/s	longitudinal wheel slip velocity
<pre>.slip_velocity_lateral	m/s	lateral wheel slip velocity
<pre>.compression_velocity_vertical	m/s	tyre compression velocity
<pre>.angular_velocity_yaw	m/s	tyre yaw velocity
<pre>.contact_point_coordinate_x	m	global x coordinate contact point
<pre>.contact_point_coordinate_y	m	global y coordinate contact point
<pre>.contact_point_coordinate_z	m	global z coordinate contact point
<pre>.road_normal_component_x	-	global x component road normal
<pre>.road_normal_component_y	-	global y component contact point
<pre>.road_normal_component_z	-	global z component contact point
<pre>.effective_road_height	m	effective road height
<pre>.effective_road_slope	rad	effective forward slope
<pre>.effective_road_curvature	1/m	effective road curvature
<pre>.effective_road_banking	rad	effective road banking/road camber angle

24.10.5 Tire Model FTire

Name UAQ	Unit	Info (data type, if not integer)
<pre> := FTire.<pos>		
<pos> := FL, FR, RL, RR or FL.Twin, FR.Twin, RL.Twin, RR.Twin		
<pre>.RoadTemp	K	road surface temperature
<pre>.PressureSetFromExt	bar	current ‘cold’ inflation pressure from extern (-99999: no change in pressure)
<pre>.P_x	m	geometrical center of contact patch in global coordinates
<pre>.P_y	m	geometrical center of contact patch in global coordinates
<pre>.P_z	m	geometrical center of contact patch in global coordinates
<pre>.tire_defelction	m	tire deflection

Name UAQ	Unit	Info (data type, if not integer)
<pre>.C_v_y	m/s	vertical rim center velocity
<pre>.C_v_x	m/s	longitudinal rim center velocity
<pre>.P_v_longslip	m/s	longitudinal rim center velocity
<pre>.P_v_latslip	m/s	lateral slip velocity at contact point
<pre>.belt_rmax_inflation	m	maximum belt radius after inflation
<pre>.rim_rotv_2WC	rad/s	rim angular velocity relative to wheel carrier (ABS signal)
<pre>.friction	-	mean road friction factor in contact patch
<pre>.Frc_C	N	tire forces expressed in TYDEX C-axis
<pre>.Trq_C	Nm	tire torques expressed in TYDEX C-axis
<pre>.rollingloss	N	rolling loss
<pre>.belt2rim_intrusion	m	maximum belt-to-rim contact intrusion
<pre>.dynradius	m	dynamic rolling radius
<pre>.Frc_ISO	N	tire forces expressed in ISO axis
<pre>.Trq_ISO	Nm	tire torques expressed in ISO axis

24.11 Brake

24.11.1 Brake - General

Name UAQ	Name C-Code	Name CM4SL	Unit	Info (data type, if not double)
<preIF> := Brake.IF				
Brake.Park	<preIF>.Park	-	-	Park brake actuation (0..1)
Brake.Pedal	<preIF>.Pedal	-	-	Brake pedal actuation (0..1)
Brake.Trq_DriveSrc_trg_<*ds*>	<preIF>.Trq_DriveSrc_trg[*ds*]	Brake IF Trq_DriveSrc_trg d<ds>	Nm	Optional target torque at drive source *ds* (e.g. by ESP ECU)
Brake.Trq_Reg_trg_FL Brake.Trq_Reg_trg_FR Brake.Trq_Reg_trg_RL Brake.Trq_Reg_trg_RR	<preIF>.Trq_Reg_trg[0] <preIF>.Trq_Reg_trg[1] <preIF>.Trq_Reg_trg[2] <preIF>.Trq_Reg_trg[3]	Brake IF Trq_Reg_trg FL Brake IF Trq_Reg_trg FR Brake IF Trq_Reg_trg RL Brake IF Trq_Reg_trg RR	Nm	Target regenerative brake torque front left / front right / rear left / rear right
Brake.Trq_FL_ext Brake.Trq_FR_ext Brake.Trq_RL_ext Brake.Trq_RR_ext	Brake.Trq_ext[0] Brake.Trq_ext[1] Brake.Trq_ext[2] Brake.Trq_ext[3]	-	Nm	Additional external brake torque front left / front right / rear left / rear right
Brake.Trq_FL_tot Brake.Trq_FR_tot Brake.Trq_RL_tot Brake.Trq_RR_tot	Brake.Trq_tot[0] Brake.Trq_tot[1] Brake.Trq_tot[2] Brake.Trq_tot[3]	-	Nm	Total brake torque (Trq_WB + Trq_PB + Trq_ext) front left / front right / rear left / rear right
Brake.Trq_PB_FL Brake.Trq_PB_FR Brake.Trq_PB_RL Brake.Trq_PB_RR	<preIF>.Trq_PB[0] <preIF>.Trq_PB[1] <preIF>.Trq_PB[2] <preIF>.Trq_PB[3]	Brake IF Trq_PB FL Brake IF Trq_PB FR Brake IF Trq_PB FR Brake IF Trq_PB FR	Nm	Brake torque of park brake front left / front right / rear left / rear right
Brake.Trq_WB_FL Brake.Trq_WB_FR Brake.Trq_WB_RL Brake.Trq_WB_RR	<preIF>.Trq_WB[0] <preIF>.Trq_WB[1] <preIF>.Trq_WB[2] <preIF>.Trq_WB[3]	Brake IF Trq_WB FL Brake IF Trq_WB FR Brake IF Trq_WB FL Brake IF Trq_WB RR	Nm	Brake torque at wheel brake front left / front right / rear left / rear right

24.11.2 Brake - Hydraulic

Hydraulic Brake Control - General

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
Brake.Hyd.CU.BooSignal	Brake.HydBrakeCU_IF.BooSignal	-	Booster activation signal (0..1)
Brake.Hyd.CU.Park	Brake.HydBrakeCU_IF.Park	-	Park brake actuation (0..1)
Brake.Hyd.CU.Pedal	Brake.HydBrakeCU_IF.Pedal	-	Brake pedal actuation (0..1)
Brake.Hyd.CU.PumpCtrl	Brake.HydBrakeCU_IF.PumpCtrl	-	Hydraulic pump activated (boolean)
Brake.Hyd.CU.Valve_In_FL Brake.Hyd.CU.Valve_In_FR Brake.Hyd.CU.Valve_In_RL Brake.Hyd.CU.Valve_In_RR	Brake.HydBrakeCU_IF.V[0-3]	-	Valve activity for inlet valve front left / front right / rear left / rear right (0..1)
Brake.Hyd.CU.Valve_Out_FL Brake.Hyd.CU.Valve_Out_FR Brake.Hyd.CU.Valve_Out_RL Brake.Hyd.CU.Valve_Out_RR	Brake.HydBrakeCU_IF.V[4-7]	-	Valve activity for outlet valve front left/ front right/ rear left / rear right (0..1)
Brake.Hyd.CU.Valve_PV_0	Brake.HydBrakeCU_IF.V[8]	-	Valve activity for pilot valve 0
Brake.Hyd.CU.Valve_PV_1	Brake.HydBrakeCU_IF.V[9]	-	Valve activity for pilot valve 1
Brake.Hyd.CU.Valve_SV_0	Brake.HydBrakeCU_IF.V[10]	-	Valve activity for suction valve 0
Brake.Hyd.CU.Valve_SV_1	Brake.HydBrakeCU_IF.V[11]	-	Valve activity for suction valve 1

Hydraulic Brake Control - HydBasic

Name UAQ	Unit	Info (data type, if not double)
Brake.Hyd.CU.Basic.desiredTrq_<pos>	Nm	Desired total (regenerative and conventional) static brake torque for the wheel <pos>

Name UAQ	Unit	Info (data type, if not double)
Brake.Hyd.CU.Basic.dTrq_<pos>	Nm	Torque difference between the desired and current total brake torque
Brake.Hyd.CU.Basic.regCover_<pos>	%	The coverage ratio of the regenerative brake torque to the total brake torque

Hydraulic Brake System - General

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
Brake.Hyd.Sys.DiaphTravel	Brake.HydBrakeIF.DiaphTravel	m	Travel of the booster diaphragm
Brake.Hyd.Sys.PedFrc	Brake.HydBrakeIF.PedFrc	N	Force applied on brake pedal
Brake.Hyd.Sys.PedTravel	Brake.HydBrakeIF.PedTravel	m	Brake pedal travel
Brake.Hyd.Sys.PistTravel	Brake.HydBrakeIF.PistTravel	m	Travel of master cylinder brake piston
Brake.Hyd.Sys.pMC	Brake.HydBrakeIF.pMC	bar	Master cylinder pressure
Brake.Hyd.Sys.pMC_in	Brake.HydBrakeIF.pMC_in	bar	Master cylinder pressure to hydraulic model for input mode 'Use_pMCInput' (input from file)
Brake.Hyd.Sys.PuRetVolt	Brake.HydBrakeIF.PuRetVolt	V	Hydraulic pump return voltage
Brake.Hyd.Sys.pWB_FL Brake.Hyd.Sys.pWB_FR Brake.Hyd.Sys.pWB_RL Brake.Hyd.Sys.pWB_RR	Brake.HydBrakeIF.pWB[0-3]	bar	Brake pressure at wheel base front left / front right / rear left / rear right
Brake.Hyd.Sys.Rel_SW	Brake.HydBrakeIF.Rel_SW	-	Brake booster release switch actuated (boolean)
Brake.Hyd.Sys.Use_pMCInput	Brake.HydBrakeIF.Use_pMCInput	-	Flag indicating whether Brake.pMC_in (pressure in master cylinder) is taken into account instead of pedal position for input from file (boolean)

Hydraulic Brake System - HydESP

Name UAQ	Unit	Info (data type, if not integer)
Brake.Hyd.Sys.ESP.Att_0.p	bar	Pressure of attenuator 0
Brake.Hyd.Sys.ESP.Att_0.v	m^3	Volume of attenuator 0
Brake.Hyd.Sys.ESP.Att_1.p	bar	Pressure of attenuator 1
Brake.Hyd.Sys.ESP.Att_1.v	m^3	Volume of attenuator 1
Brake.Hyd.Sys.ESP.Booster.F	N	Booster output force (input to master cylinder piston)
Brake.Hyd.Sys.ESP.Cyl_FL.p Brake.Hyd.Sys.ESP.Cyl_FR.p Brake.Hyd.Sys.ESP.Cyl_RL.p Brake.Hyd.Sys.ESP.Cyl_RR.p	bar	Pressure of brake cylinder front left front right rear left rear right
Brake.Hyd.Sys.ESP.Cyl_FL.v Brake.Hyd.Sys.ESP.Cyl_FR.v Brake.Hyd.Sys.ESP.Cyl_RL.v Brake.Hyd.Sys.ESP.Cyl_RR.v	m^3	Volume of brake cylinder front left front right rear left rear right
Brake.Hyd.Sys.ESP.In_FL.alpha Brake.Hyd.Sys.ESP.In_FR.alpha Brake.Hyd.Sys.ESP.In_RL.alpha Brake.Hyd.Sys.ESP.In_RR.alpha	-	Normalized opening of inlet valve front left front right rear left rear right
Brake.Hyd.Sys.ESP.IN_FL.dp Brake.Hyd.Sys.ESP.IN_FR.dp Brake.Hyd.Sys.ESP.IN_RL.dp Brake.Hyd.Sys.ESP.IN_RR.dp	bar	Pressure difference of inlet valve front left front right rear left rear right
Brake.Hyd.Sys.ESP.IN_FL.q Brake.Hyd.Sys.ESP.IN_FR.q Brake.Hyd.Sys.ESP.IN_RL.q Brake.Hyd.Sys.ESP.IN_RR.q	m^3/s	Volume flow through inlet valve front left front right rear left rear right
Brake.Hyd.Sys.ESP.In_FL.state Brake.Hyd.Sys.ESP.In_FR.state Brake.Hyd.Sys.ESP.In_RL.state Brake.Hyd.Sys.ESP.In_RR.state	-	State of inlet valve (integer) front left front right rear left rear right 0=not switchable, 1=switched to great, 2=switched to small
Brake.Hyd.Sys.ESP.LPA_0.p	bar	Pressure of low pressure accumulator 0
Brake.Hyd.Sys.ESP.LPA_0.v	m^3	Volume of low pressure accumulator 0

Name UAQ	Unit	Info (data type, if not integer)
Brake.Hyd.Sys.ESP.LPA_1.p	bar	Pressure of low pressure accumulator 1
Brake.Hyd.Sys.ESP.LPA_1.v	m^3	Volume of low pressure accumulator 1
Brake.Hyd.Sys.ESP.nPump	1/s	Rotation speed of hydraulic pump
Brake.Hyd.Sys.ESP.Out_FL.alpha Brake.Hyd.Sys.ESP.Out_FR.alpha Brake.Hyd.Sys.ESP.Out_RL.alpha Brake.Hyd.Sys.ESP.Out_RR.alpha	-	Normalized opening of outlet valve front left front right rear left rear right
Brake.Hyd.Sys.ESP.OUT_FL.dp Brake.Hyd.Sys.ESP.OUT_FR.dp Brake.Hyd.Sys.ESP.OUT_RL.dp Brake.Hyd.Sys.ESP.OUT_RR.dp	bar	Pressure difference of outlet valve front left front right rear left rear right
Brake.Hyd.Sys.ESP.OUT_FL.q Brake.Hyd.Sys.ESP.OUT_FR.q Brake.Hyd.Sys.ESP.OUT_RL.q Brake.Hyd.Sys.ESP.OUT_RR.q	m^3/s	Volume flow through outlet valve front left front right rear left rear right
Brake.Hyd.Sys.ESP.Out_FL.state Brake.Hyd.Sys.ESP.Out_FR.state Brake.Hyd.Sys.ESP.Out_RL.state Brake.Hyd.Sys.ESP.Out_RR.state	-	State of outlet valve (integer) front left front right rear left rear right 0=not switchable, 1=switched to great, 2=switched to small
Brake.Hyd.Sys.ESP.Pu_0.dp	bar	Pressure difference of hydraulic pump 0
Brake.Hyd.Sys.ESP.Pu_0.q	m^3/s	Volume flow through hydraulic pump 0
Brake.Hyd.Sys.ESP.Pu_1.dp	bar	Pressure difference of hydraulic pump 1
Brake.Hyd.Sys.ESP.Pu_1.q	m^3/s	Volume flow through hydraulic pump 1
Brake.Hyd.Sys.ESP.Pump.dp	bar	Pressure difference of hydraulic pump
Brake.Hyd.Sys.ESP.PV_0.alpha	-	Normalized opening of pilot valve 0
Brake.Hyd.Sys.ESP.PV_0.dp	bar	Pressure difference of pilot valve 0
Brake.Hyd.Sys.ESP.PV_0.q	m^3/s	Volume flow through pilot valve 0
Brake.Hyd.Sys.ESP.PV_0.state	-	State of pilot valve 0 (integer) 0=not switchable, 1=switched to great, 2=switched to small
Brake.Hyd.Sys.ESP.PV_1.alpha	-	Normalized opening of pilot valve 1
Brake.Hyd.Sys.ESP.PV_1.dp	bar	Pressure difference of pilot valve 1
Brake.Hyd.Sys.ESP.PV_1.q	m^3/s	Volume flow through pilot valve 1
Brake.Hyd.Sys.ESP.PV_1.state	-	State of pilot valve 1 (integer) 0=not switchable, 1=switched to great, 2=switched to small
Brake.Hyd.Sys.ESP.SuppL_0.p	bar	Pressure of supply line 0
Brake.Hyd.Sys.ESP.SuppL_0.v	m^3	Volume of supply line 0
Brake.Hyd.Sys.ESP.SuppL_1.p	bar	Pressure of supply line 1
Brake.Hyd.Sys.ESP.SuppL_1.v	m^3	Volume of supply line 1
Brake.Hyd.Sys.ESP.SV_0.alpha	-	Normalized opening of suction valve 0
Brake.Hyd.Sys.ESP.SV_0.dp	bar	Pressure difference of suction valve 0
Brake.Hyd.Sys.ESP.SV_0.q	m^3/s	Volume flow through suction valve 0
Brake.Hyd.Sys.ESP.SV_0.state	-	State of suction valve 0 (integer) 0=not switchable, 1=switched to great, 2=switched to small
Brake.Hyd.Sys.ESP.SV_1.alpha	-	Normalized opening of suction valve 1
Brake.Hyd.Sys.ESP.SV_1.dp	bar	Pressure difference of suction valve 1
Brake.Hyd.Sys.ESP.SV_1.q	m^3/s	Volume flow through suction valve 1
Brake.Hyd.Sys.ESP.SV_1.state	-	State of suction valve 1 (integer) 0=not switchable, 1=switched to great, 2=switched to small

24.12 Powertrain

24.12.1 General

Name UAQ	Name C-Code	Name CM4SL	Unit	Info (data type, if not double)
<pos> := FL, FR, RL, RR <pre> := PowerTrain <Nb> :=0, .., n (number) <preW> := PowerTrain.IF.WheelOut[Nb]				
PT.OSRate	<pre>.OSRate		-	Oversampling rate (integration substeps)

PT.Trq_Supp2Bdy1B.y	<pre>.Trq_Supp2Bdy1B[0]
	<pre>.Trq_Supp2Bdy1B[1]
	<pre>.Trq_Supp2Bdy1B[2]
		PT IF Trq_Supp2Bdy1B x
		PT IF Trq_Supp2Bdy1B y
		PT IF Trq_Supp2Bdy1B z
			Nm	Support torque to vehicle frame Fr1B

PT.Trq_Supp2Bdy1.y	<pre>.Trq_Supp2Bdy1[0]
	<pre>.Trq_Supp2Bdy1[1]
	<pre>.Trq_Supp2Bdy1[2]
		PT IF Trq_Supp2Bdy1 x
		PT IF Trq_Supp2Bdy1 y
		PT IF Trq_Supp2Bdy1 z
			Nm	Support torque to vehicle frame Fr1/Fr1A

PT.Trq_Supp2BdyEng.x
PT.Trq_Supp2BdyEng.y
	<pre>.Trq_Supp2BdyEng[0]
	<pre>.Trq_Supp2BdyEng[1]
		PT IF Trq_Supp2BdyEng x
		PT IF Trq_Supp2BdyEng y
			Nm	Support torque to engine frame FrEng

PT.W<pos>.rot	<preW>.rot	PT IF W<pos> rot	rad	Rotation angle of wheel <pos>
PT.W<pos>.rotv	<preW>.rotv	PT IF W<pos> rotv	rad/s	Rotation speed of wheel <pos>
PT.W<pos>.Trq_B2W	<preW>.Trq_B2W	PT IF W<pos> Trq_B2W	Nm	Torque acting from brake to wheel <pos>
PT.W<pos>.Trq_BrakeReg	<preW>.Trq_BrakeReg	PT IF W<pos> Trq_BrakeReg	Nm	Estimated current regenerative brake torque at wheel <pos> (absolute value)
PT.W<pos>.Trq_BrakeReg_max	<preW>.Trq_BrakeReg_max	PT IF W<pos> Trq_BrakeReg_max	Nm	Estimated maximum possible regenerative brake torque at wheel <pos> (absolute value)
PT.W<pos>.Trq_Drive	<preW>.Trq_Drive	PT IF W<pos> Trq_Drive	Nm	Drive torque of wheel <pos>
PT.W<pos>.Trq_Supp2WC	<preW>.Trq_Supp2WC	PT W<pos>Trq_Supp2WC	Nm	Drive torque support on wheel carrier <pos>

24.12.2 Powertrain Control (PTControl)

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<pre> := PT.Control <preGBM> := PT.Contro.GB_M<iM> <preIF> := PowerTrain.ControlIF <preIFGB> := PowerTrain.ControlIF.GearBoxOut <preIFGBM> := PowerTrain.ControlIF.GearBoxM_Out[iM]			
<pre>.Clutch.Pos	<preIF>.ClutchOut.Pos	-	Clutch target position
<pre>.Clutch.rotv_out_trg	<preIF>.ClutchOut.rotv_out_trg	rad/s	Clutch output shaft target rotation speed
<pre>.Clutch.Trq_out_trg	<preIF>.ClutchOut.Trq_out_trg	Nm	Clutch output shaft target torque

<pre>.Consump.Comb_E.Abs
<pre>.Consump.Comb_E.Act
<pre>.Consump.Comb_E.Avg
		kWh
		kWh/100km
			Combined consumption (fuel, low and high voltage battery) expressed as electric energy consumption: absolute, actual, average

<pre>.Consump.Comb_F.Abs
<pre>.Consump.Comb_F.Act
<pre>.Consump.Comb_F.Avg
		l
		l/100km
			Combined consumption (fuel, low and high voltage battery) expressed as fuel consumption: absolute, actual, average

<pre>.Consump.Elec.Abs
<pre>.Consump.Elec.Act
<pre>.Consump.Elec.Avg
		kWh
		kWh/100km
			Electric energy consumption (low and high voltage battery): absolute, actual, average

<pre>.Consump.Fuel.Abs
<pre>.Consump.Fuel.Act
<pre>.Consump.Fuel.Avg
		l
		l/100km
			Fuel consumption: absolute, actual, average

<pre>.Consump.Reset		-	Reset the absolute and average consumption values (boolean)
<pre>.Engine.FuelCutOff	<preIF>.EngineOut.FuelCutOff	-	Flag if fuel is cut-off (boolean)
<pre>.Engine.Load	<preIF>.EngineOut.Load	-	Engine target load
<pre>.Engine.rotv_trg	<preIF>.EngineOut.rotv_trg	rad/s	Engine target rotation speed
<pre>.Engine.set_ISC	<preIF>.EngineOut.set_ISC	-	Idle speed controller activated (boolean)
<pre>.Engine.Trq_trg	<preIF>.EngineOut.Trq_trg	Nm	Engine target torque
<pre>.GasInterpret.Trq_trg		Nm	Target drive torque from gas pedal interpreter module

<pre>.GB.Clutch.Pos
<preGBM>.Clutch.Pos
	<preIFGB>.Clutch.Pos
	<preIFGBM>.Clutch.Pos
		-	Gearbox internal clutch target position

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<pre>.GB.Clutch_dis.Pos	<preIFGB>.Clutch_dis.Pos	-	Gearbox internal clutch target position for disengaged shaft of DCT gearbox

<pre>.GB.Clutch.rotv_out_trg
<preGBM>.Clutch.rotv_out_trg
	<preIFGB>.Clutch.rotv_out_trg
	<preIFGBM>.Clutch.rotv_out_trg
		rad/s	Gearbox internal clutch output shaft target rotation speed

<pre>.GB.Clutch_dis.rotv_out_trg	<preIFGB>.Clutch_dis.rotv_out_trg	rad/s	Gearbox internal clutch output shaft target rotation speed for disengaged shaft of DCT gearbox

<pre>.GB.Clutch.Trq_out_trg
<preGBM>.Clutch.Trq_out_trg
	<preIFGB>.Clutch.Trq_out_trg
	<preIFGBM>.Clutch.Trq_out_trg
		Nm	Gearbox internal clutch output shaft target torque

<pre>.GB.Clutch_dis.Trq_out_trg	<preIFGB>.Clutch_dis.Trq_out_trg	Nm	Gearbox internal clutch output shaft target torque for disengaged shaft of DCT gearbox

<pre>.GB.GearNoTrg
<preGBM>.GearNoTrg
	<preIFGB>.GearNoTrg
	<preIFGBM>.GearNoTrg
		-	Gearbox target gear (integer)

<pre>.GB.GearNoTrg_dis	<preIFGB>.GearNoTrg_dis	-	Gearbox target gear (integer) for disengaged shaft of DCT gearbox

<pre>.GB.rotv_in_trg
<preGBM>.rotv_in_trg
	<preIFGB>.rotv_in_trg
	<preIFGBM>.rotv_in_trg
		rad/s	Gearbox input shaft target rotation speed

<pre>.GB.Trq_out_trg
<preGBM>.Trq_out_trg
	<preIFGB>.Trq_out_trg
	<preIFGBM>.Trq_out_trg
		Nm	Gearbox output shaft target torque

<pre>.Ignition	<preIF>.Ignition	-	Powertrain ignition (boolean)

<pre>.ISG.Load
<pre>.Motor<iM>.Load
	<preIF>.ISGOut.Load
	<preIF>.MotorOut[iM].Load
		-	Integrated starter generator / electric motor <iM> target load

<pre>.ISG.rotv_trg
<pre>.Motor<iM>.rotv_trg
	<preIF>.ISGOut.rotv_trg
	<preIF>.MotorOut[iM].rotv_trg
		rad/s	Integrated starter generator / electric motor <iM> target rotation speed

<pre>.ISG.Trq_trg
<pre>.Motor<iM>.Trq_trg
	<preIF>.ISGOut.Trq_trg
	<preIF>.MotorOut[iM].Trq_trg
		Nm	Integrated starter generator / electric motor <iM> target torque

<pre>.OperationError	<preIF>.OperationError	-	Current operation error(integer)
<pre>.OperationState	<preIF>.OperationState	-	Current operation state (integer): 0: Absent (leave the vehicle) 1: Power off 2: Power accessory 3: Power on 4: Driving
<pre>.PwrExploited		%	Ratio of exploited power (for visualization in IPGInstruments)

<pre>.PwrSupply.
Pwr_HV1toLV_trg
	<preIF>.PwrSupplyOut.
	Pwr_HV1toLV_trg
		W	Target transferred electric power from high voltage 1 electric circuit to low voltage electric circuit

<pre>.StrategyMode	<preIF>.StrategyMode	-	Current strategy mode (integer): 0: Start/Stop 1: Regenerative brake 2: Regenerative drag torque 3: Coasting 4: Electric drive 5: LoadShift 6: Assist 7: Boost 8: Engine drive 9: Engine start 10:Engine synchronization 11: Engine stop
<pre>.StrategyMode_trg		-	Target strategy mode (integer)

<pre>.StrategyMode_trg.
SetManual
		-	Flag if target strategy mode can be set by DVA (boolean):
			1=target strategy mode is set manually by DVA and not by PTControl

<pre>.TrqRatio_f			For electrical and serial powertrain the ratio for torque distribution between the front and rear drive axle (0=all at rear axle....1=all at front axle).

24.12.3 Engine Control Unit (ECU)

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<pre> := PowerTrain.EngineCU_IF			
PT.ECU.Engine_on	<pre>.Engine_on	-	Flag if engine is on (boolean)
PT.ECU.FuelCutOff	<pre>.FuelCutOff	-	Flag if fuel is cut-off (boolean)
PT.ECU.Load	<pre>.Load	-	Load/throttle signal for engine
PT.ECU.Status	<pre>.Status	-	ECU status (integer)
PT.ECU.TrqDrag	<pre>.TrqDrag	Nm	Engine drag torque at current rotation speed
PT.ECU.TrqFull	<pre>.TrqFull	Nm	Engine full torque at current rotation speed
PT.ECU.TrqOpt	<pre>.TrqOpt	Nm	Engine torque with optimum consumption at current rotation speed
PT.ECU.Trq_lim		Nm	Target engine torque limitation (only for model *Basic*)
PT.ECU.Load_lim			Output engine load limitation (only for model *Basic*)

24.12.4 Motor Control Unit (MCU)

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<pre> := PowerTrain.MotorCU_IF <preM> := PT.MCU.Motor<iM>			

PT.MCU.ISG.Load
<preM>.Load
	<pre>.ISGOut.Load
	<pre>.MotorOut[iM].Load
		-	Load signal for integrated starter generator / electric motor <iM>

PT.MCU.ISG.TrqGen_max
<preM>.TrqGen_max
	<pre>.ISGOut.TrqGen_max
	<pre>.MotorOut[iM].TrqGen_max
		Nm	Integrated starter generator / electric motor <iM> maximum generator torque at current rotation speed

PT.MCU.ISG.TrqMot_max
<preM>.TrqMot_max
	<pre>.ISGOut.TrqMot_max
	<pre>.MotorOut[iM].TrqMot_max
		Nm	Integrated starter generator / electric motor <iM> maximum motor torque at current rotation speed

PT.MCU.Status	<pre>.Status	-	MCU status (integer)

24.12.5 Transmission Control Unit (TCU)

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<pre> := PowerTrain.TransmCU_IF <preGB> := PT.TCU.GB <preGB_M> := PT.TCU.GB_M<iM>			
PT.TCU.Clutch.Pos	<pre>.ClutchOut.Pos	-	Separated clutch target position

<preGB>.Clutch.Pos
<preGB_M>.Clutch.Pos
	<pre>.GearBoxOut.Clutch.Pos
	<pre>.GearBoxM_Out[iM].Clutch.Pos
		-	Internal clutch target position of gearbox and electric motor gearbox <iM> (0 = clutch closed, 1 = clutch open)

<preGB>.Clutch_dis.Pos	<pre>.GearBoxOut.Clutch_dis.Pos	-	Internal clutch target position of gearbox for disengaged shaft of DCT

<preGB>.GearNoTrg
<preGB_M>.GearNoTrg
	<pre>.GearBoxOut.GearNoTrg
	<pre>.GearBoxM_Out[iM].GearNoTrg
		-	Target gear of gearbox and electric motor gearbox <iM>

<preGB>.GearNoTrg_dis	<pre>.GearBoxOut.GearNoTrg_dis	-	Target gear of gearbox for disengaged shaft of DCT gearbox
<preGB>.i_trg	<pre>.GearBoxOut.i_trg	-	Target gear ratio for CVT gearbox

<preGB>.set_ParkBrake
<preGB_M>.set_ParkBrake
	<pre>.GearBoxOut.set_ParkBrake
	<pre>.GearBoxM_Out[iM].set_ParkBrake
		-	Set gearbox park brake of gearbox and electric motor gearbox <iM>

<preGB>.Trq_DriveSrc_trg
<preGB_M>.Trq_DriveSrc_trg
	<pre>.GearBoxOut.Trq_DriveSrc_trg
	<pre>.GearBoxM_Out[iM].Trq_DriveSrc_trg
		Nm	Optional drive source target torque from TCU to PTControl (e.g. while shifting)

PT.TCU.LockUp		-	Lock-up clutch position of gearbox converter clutch with TCU “Automatic” or “Automatic with Converter“ (0 = no lockup, 1 = lockup clutch closed)
PT.TCU.Status	<pre>.Status	-	TCU status (integer)

24.12.6 Battery Control Unit (BCU)

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<pre> := PowerTrain.BatteryCU_IF			
PT.BCU.SOC_HV	<pre>.SOC_HV	%	High voltage battery actual state of charge
PT.BCU.SOC_LV	<pre>.SOC_LV	%	Low voltage battery actual state of charge
PT.BCU.SOH_HV	<pre>.SOH_HV	%	High voltage battery state of health
PT.BCU.SOH_LV	<pre>.SOH_LV	%	Low voltage battery state of health
PT.BCU.Status	<pre>.Status	-	BCU status (integer)

24.12.7 Engine

General

Name UAQ	Name C-Code	Name CM4SL	Unit	Info (data type, if not double)
<pre> := PowerTrain.Engine <preIF> := PowerTrain.EngineIF				
		PT IF Engine_Trq (Engine Control)	Nm	Engine control torque
PT.Engine.be			g/kWh	Specific fuel consumption of the engine
PT.Engine.FuelFlow	<preIF>.FuelFlow		l/s	Current fuel flow
PT.Engine.FuelFlow_ext	<pre>.Fuel.Flow_ext		l/s	External fuel flow (negative for fill up the fuel tank)
PT.Engine.FuelLevel	<pre>.Fuel.Level		%	Current fuel tank level
PT.Engine.DVA.Trq PT.Engine.Sherpa.Trq			Nm	Effective engine torque at output shaft used for engine model 'DVA' respectively 'Sherpa'
PT.Engine.PwrO			W	Engine output power
PT.Engine.rot	<preIF>.rot		rad	Rotation angle of engine output shaft
PT.Engine.rotv	PowerTrain.IF.Engine_rotv (copy from <preIF>.rotv)	PT IF Engine_rotv	rad/s	Rotational speed of engine output shaft
PT.Engine.Trq	<preIF>.Trq		Nm	Effective engine torque at output shaft
PT.Engine.Trq_Ext2E	<pre>.Trq_Ext2E		Nm	External torque to engine output shaft
PT.Engine.Mapping.Pwr-Corr.Fac			-	Engine power/torque correction factor depending on environment conditions

Intake Manifold Pressure

Name UAQ	Name C-Code	Name CM4SL	Unit	Info (data type, if not double)
<pre> := PowerTrain.Engine.Mapping.IMP				
<pre>.dm_cyl			kg/s	Air mass flow in the engine cylinders
<pre>.dm_thr			kg/s	Air mass flow after throttle
<pre>.dpres_im			Pa/s	Intake manifold pressure derivation
<pre>.alpha			-	Throttle loss coefficient
<pre>.lambda			-	Intake manifold relative filling
<pre>.psi			-	Flow coefficient
<pre>.pres_im			Pa	Intake manifold pressure
<pre>.relrho_im			-	Intake manifold relative pressure (used for Linear2D)

24.12.8 Integrated Starter Generator / Electric Motor

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
PT.MotorISG.PwrElec PT.Motor<iM>.PwrElec	PowerTrain.ISG_IF.PwrElec PowerTrain.MotorIF[iM].PwrElec	W	Actual electric power of integrated starter generator / electric motor <iM>
PT.MotorISG.rot PT.Motor<iM>.rot	PowerTrain.ISG_IF.rot PowerTrain.MotorIF[iM].rot	rad	Actual output shaft rotation angle of integrated starter generator / electric motor <iM>
PT.MotorISG.rotv PT.Motor<iM>.rotv	PowerTrain.ISG_IF.rotv PowerTrain.MotorIF[iM].rotv	rad/s	Actual output shaft rotation speed of integrated starter generator / electric motor <iM>
PT.MotorISG.Trq PT.Motor<iM>.Trq	PowerTrain.ISG_IF.Trq PowerTrain.MotorIF[iM].Trq	Nm	Actual torque on driven shaft of integrated starter generator / electric motor <iM>
PT.MotorISG.Voltage PT.Motor<iM>.Voltage	PowerTrain.ISG_IF.Voltage PowerTrain.MotorIF[iM].Voltage	V	Electric voltage at integrated starter generator / electric motor <iM>

24.12.9 Clutch

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<pre> := PowerTrain.ClutchIF			
PT.Clutch.i_TrqIn2Out	<pre>.i_TrqIn2Out	-	Current ratio clutch input shaft torque to output shaft torque (estimated)
PT.Clutch.DVA.Trq_A2B		Nm	Torque transferred from clutch input to output side used for clutch model ‘DVA’
PT.Clutch.PwrI		W	Clutch input power
PT.Clutch.PwrO		W	Clutch output power
PT.Clutch.rot_in	<pre>.rot_in	rad	Rotation angle of clutch input shaft
PT.Clutch.rot_out	<pre>.rot_out	rad	Rotation angle of clutch output shaft
PT.Clutch.rotv_in	<pre>.rotv_in	rad/s	Rotational speed of clutch input shaft
PT.Clutch.rotv_out	<pre>.rotv_out	rad/s	Rotational speed of clutch output shaft
PT.Clutch.Trq_in	<pre>.Trq_in	Nm	Clutch input shaft torque
PT.Clutch.Trq_out	<pre>.Trq_out	Nm	Clutch output shaft torque

24.12.10 GearBox

Name UAQ	Name C-Code	Name CM4SL	Unit	Info (data type, if not double)
<pre> := PT.GearBox <preM> := PT.GearBoxM<iM> <preIF> := PowerTrain.GearBoxIF <preIF_M> := PowerTrain.GearBoxM_IF[iM] <preGB> := PowerTrain.GearBox				
<pre>.DVA.i			-	Current user defined transmission ratio used for gearbox model 'DVA'

<pre>.Clutch.i_TrqIn2Out
<preM>.Clutch.i_TrqIn2Out
	<preIF>.ClutchOut.i_TrqIn2Out
	<preIF_M>.ClutchOut.i_TrqIn2Out
			-	Ratio between gearbox clutch input shaft and output shaft (estimated)

<pre>.Clutch_dis.i_TrqIn2Out	<preIF>.Clutch_dis_Out.i_TrqIn2Out		-	Ratio between gearbox clutch input shaft and output shaft (estimated) for disengaged shaft of DCT gearbox

<pre>.Clutch.rotv_in
<preM>.Clutch.rotv_in
	<preIF>.ClutchOut.rotv_in
	<preIF_M>.ClutchOut.rotv_in
			rad/s	Current gearbox clutch input shaft rotation angle

<pre>.Clutch_dis.rotv_in	<preIF>.Clutch_dis_Out.rotv_in		rad/s	Current gearbox clutch input shaft rotation angle for disengaged shaft of DCT gearbox

<pre>.Clutch.rotv_out
<preM>.Clutch.rotv_out
	<preIF>.ClutchOut.rotv_out
	<preIF_M>.Clutch.rotv_out
			rad/s	Current gearbox clutch output shaft rotation speed

<pre>.Clutch_dis.rotv_out	<preIF>.Clutch_dis_Out.rotv_out		rad/s	Current gearbox clutch output shaft rotation speed for disengaged shaft of DCT gearbox

<pre>.Clutch.Trq_in
<preM>.Clutch.Trq_in
	<preIF>.ClutchOut.Trq_in
	<preIF_M>.ClutchOut.Trq_in
			Nm	Current gearbox clutch input shaft torque

<pre>.Clutch_dis.Trq_in	<preIF>.Clutch_dis_Out.Trq_in		Nm	Current gearbox clutch input shaft torque for disengaged shaft of DCT gearbox

<pre>.Clutch.Trq_out
<preM>.Clutch.Trq_out
	<preIF>.ClutchOut.Trq_out
	<preIF_M>.ClutchOut.Trq_out
			Nm	Current gearbox clutch output shaft torque

<pre>.Clutch_dis.Trq_out	<preIF>.Clutch_dis_Out.Trq_out		Nm	Current gearbox clutch output shaft torque for disengaged shaft of DCT gearbox
<pre>.CVT.didt			1/s	Current gearbox ratio change rate for gearbox model CVT
<pre>.CVT.efficiency			-	Current gearbox efficiencyfor gearbox model CVT

<pre>.GearNo
 	
<preM>.GearNo
	PowerTrain.IF.GearNo
	(copy from <pre>.GearNo)
	<preIF_M>.GearNo
		PT IF GearNo	-	Current gear (integer)

<pre>.GearNo_dis	<preIF>.GearNo_dis	PT IF GearNo_dis	-	Current gear of disengaged shaft (integer) for DCT

<pre>.i
<preM>.i
	<preIF>.i
	<preIF_M>.i
			-	Current transmission ratio

<pre>.i_Ext	<preGB>.i_Ext		-	Additional user defined external transmission ratio

<pre>.i_TrqIn2Out
<preM>.i_TrqIn2Out
	<preIF>.i_TrqIn2Out
	<preIF_M>.i_TrqIn2Out
			-	Current gearbox transmission ratio considering loss

<pre>.Inert_ExtIn	<preGB>.Inert_ExtIn		kgm^2	Additional external inertia to gearbox input shaft
<pre>.Inert_ExtOut	<preGB>.Inert_ExtOut		kgm^2	Additional external inertia to gearbox output shaft
<pre>.Inert_out	<preIF>.Inert_out		kgm^2	Gearbox output shaft inertia
<pre>.PwrI			W	Gearbox input power
<pre>.PwrO			W	Gearbox output power

<pre>.rot_in
<preM>.rot_in
	<preIF>.rot_in
	<preIF_M>.rot_in
			rad	Rotation angle of gearbox input shaft

<pre>.rot_out
<preM>.rot_out
	<preIF>.rot_out
	<preIF_M>.rot_out
			rad	Rotation angle of gearbox output shaft

<pre>.rotv_in
<preM>.rotv_in
	<preIF>.rotv_in
	<preIF_M>.rotv_in
			rad/s	Rotational speed of gearbox input shaft

<pre>.rotv_out
<preM>.rotv_out
	<preIF>.rotv_out
	<preIF_M>.rotv_out
			rad/s	Rotational speed of gearbox output shaft

<pre>.Trq_Ext2GB_In	<preGB>.Trq_Ext2GB_In		Nm	External torque to gearbox input shaft (in front of the internal clutch)
<pre>.Trq_Ext2GB_Out	<preGB>.Trq_Ext2GB_Out		Nm	External torque to gearbox output shaft

<pre>.Trq_in
<preM>.Trq_in
	<preIF>.Trq_in
	<preIF_M>.Trq_in
			Nm	Current gearbox input shaft torque

<pre>.Trq_out
<preM>.Trq_out
	<preIF>.Trq_out
	<preIF_M>.Trq_out
			Nm	Current gearbox output shaft torque

24.12.11 Driveline

General

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<pre> := PowerTrain.DriveLine <preIF> := PowerTrain.DriveLineIF <preDS> := PT.DL.DriveSrc.<ds> <pos> := FL, FR, RL, RR			
<preDS>.i_D2W<pos>	<preIF>.DriveOut[ds].i_D2W[pos]	-	Drive source <ds> current ratio: drive source to wheel <pos> (estimated)
<preDS>.rot_in	<preIF>.DriveOut[ds].rot_in	rad	Drive source <ds> input rotation angle
<preDS>.rotv_in	<preIF>.DriveOut[ds].rotv_in	rad/s	Drive source <ds> input rotation speed
<preDS>.Trq_in	<preIF>.DriveIn[ds].Trq_in	Nm	Drive source <ds> input torque
<preDS>.Trq_ext	<pre>.Trq_Ext2DriveSrc[ds]	Nm	Additional external input torque at outshaft of drive source <ds>
PT.DL.iDiff_mean	PowerTrain.IF.DL_iDiff_mean (copy from <preIF>.iDiff_mean)		Driveline mean differential ratio (overrides the value from the initialisation if non zero)

Generic models

Name UAQ	Unit	Info (data type, if not double)
<Diff> := FDiff, RDiff, CDiff		
PT.Gen.DL.<Diff>.DVA.Trq_A2B	Nm	Torque transferred from A to B at differential (for coupling model ‘DVA’)
PT.Gen.DL.<Diff>.PwrI	W	Differential input power
PT.Gen.DL.<Diff>.PwrO	W	Differential output power
PT.Gen.DL.<Diff>.rotv_in	rad/s	Rotation speed of differential input shaft
PT.Gen.DL.<Diff>.Trq_Cpl2B	Nm	Torque from differential coupling model to shaft B
PT.Gen.DL.<Diff>.Trq_Input	Nm	Torque to differential input shaft
PT.Gen.DL.<Diff>.TrqRatio		Torque ratio at differential used for coupling mode ‘TrqVec’
PT.Gen.DL.CDiff.GearNo		Current gear number of center differential if there exists more than one gears (integer)
PT.Gen.DL.HangOn.drotv_Diff2o	rad/s	Delta rotation speed between input and output shaft
PT.Gen.DL.HangOn.DVA.Trq_A2B	Nm	Torque transferred from A to B at hangon clutch (for coupling model ‘DVA’)
PT.Gen.DL.HangOn.PwrI	W	Hangon clutch input power
PT.Gen.DL.HangOn.PwrO	W	Hangon clutch output power
PT.Gen.DL.HangOn.Trq_Cpl2B	Nm	Torque from hangon clutch coupling model to shaft B (output)

24.12.12 Power Supply

Battery

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<pre> := PowerTrain.PowerSupplyIF			

PT.BattLV.AOC
PT.BattHV.AOC
	<pre>.BattLV.AOC
	<pre>.BattHV.AOC
		Ah	Low / high voltage battery actual amount of charge

PT.BattLV.Current
PT.BattHV.Current
	<pre>.BattLV.Current
	<pre>.BattHV.Current
		A	Low / high voltage battery electric current

PT.BattLV.Energy
PT.BattHV.Energy
	<pre>.BattLV.Energy
	<pre>.BattHV.Energy
		kWh	Low / high voltage battery remaining energy capacity

PT.BattLV.Pwr_max
PT.BattHV.Pwr_max
	<pre>.BattLV.Pwr_max
	<pre>.BattHV.Pwr_max
		W	Low / high voltage battery maximum charge/discharge power

PT.BattLV.Temp
PT.BattHV.Temp
	<pre>.BattLV.Temp
	<pre>.BattHV.Temp
		K	Low / high voltage battery temperature

PT.BattLV.Volt_oc PT.BattHV.Volt_oc		V	Actual open circuit (idle) voltage of low / high battery (with the model ‘Chen’)

Power supply

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<pre> := PowerTrain.PowerSupply <preIF> := PowerTrain.PowerSupplyIF <prePS> := PT.PwrSupply			
<prePS>.Eta_HV1toHV2	<preIF>.Eta_HV1toHV2	-	DC/DC efficiency from high voltage 1 electric circuit to high voltage 2 electric circuit
<prePS>.Eta_HV1toLV	<preIF>.Eta_HV1toLV	-	DC/DC efficiency from high voltage 1 electric circuit to low voltage electric circuit
<prePS>.HV1.Pwr <prePS>.HV2.Pwr	<preIF>.Pwr_HV1 <preIF>.Pwr_HV2	W	Total electric power (generators and consumers) on high voltage 1 / 2 electric circuit
<prePS>.LV.Pwr	<preIF>.Pwr_LV	W	Total electric power (generators and consumers) on low voltage electric circuit

<prePS>.HV1.Pwr_aux
<prePS>.HV2.Pwr_aux
	<pre>.Pwr_HV1_aux
	<pre>.Pwr_HV2_aux
		W	Additional external electric power (generators and consumers) on high voltage 1 / 2 electric circuit

<prePS>.LV.Pwr_aux	<pre>.Pwr_LV_aux	W	Additional external electric power (generators and consumers) on low voltage electric circuit
<prePS>.HV1.Voltage <prePS>.HV2.Voltage	<preIF>.Voltage_HV1 <preIF>.Voltage_HV2	V	High voltage 1 / 2 electric circuit actual voltage
<prePS>.LV.Voltage	<preIF>.Voltage_LV	V	Low voltage electric circuit actual voltage
<prePS>.Pwr_HV1toHV2	<preIF>.Pwr_HV1toHV2	W	Actual transferred electric power from high voltage 1 circuit to high voltage 2 circuit
<prePS>.Pwr_HV1toHV2_max	<preIF>.Pwr_HV1toHV2_max	W	Maximum possible transferred electric power from high voltage 1 electric circuit to high voltage 2 electric circuit
<prePS>.Pwr_HV1toLV	<preIF>.Pwr_HV1toLV	W	Actual transferred electric power from high voltage 1 circuit to low voltage circuit
<prePS>.Pwr_HV1toLV_max	<preIF>.Pwr_HV1toLV_max	W	Maximum possible transferred electric power from high voltage 1 electric circuit to low voltage electric circuit

24.13 Power Flow Calculation

24.13.1 PowerDelta

Name UAQ	Name C-Code	Unit	Info
<pos> := FL, FR, RL, RR			
PwrD.Aero.Drag	PowerDelta.Aero.Drag	W	PowerDelta due to aerodynamic drag force
PwrD.Aero.Lift	PowerDelta.Aero.Lift	W	PowerDelta due to aerodynamic lift force
PwrD.Aero.Pitch	PowerDelta.Aero.Pitch	W	PowerDelta due to aerodynamic pitch moment
PwrD.Aero.Roll	PowerDelta.Aero.Roll	W	PowerDelta due to aerodynamic roll moment
PwrD.Aero.Side	PowerDelta.Aero.Sid	W	PowerDelta due to aerodynamic side force
PwrD.Aero.Total	PowerDelta.Aero.Total	W	Total aerodynamic PowerDelta
PwrD.Aero.Yaw	PowerDelta.Aero.Yaw	W	PowerDelta due to aerodynamic yaw moment
PwrD.Brake.Total	PowerDelta.Brake_Total	W	Total brake PowerDelta
PwrD.Brake.Total<pos>	PowerDelta.Brake[i].Total	W	PowerDelta due to brake at wheel <pos>
PwrD.Grvt	PowerDelta.Grvt,	W	PowerDelta due to road inclination (gravity)
PwrD.Hitch.Frc_x PwrD.Hitch.Frc_y PwrD.Hitch.Frc_z	PowerDelta.Hitch.Frc_x PowerDelta.Hitch.Frc_y PowerDelta.Hitch.Frc_z	W	PowerDelta due to hitch forces
PwrD.Hitch.Total	PowerDelta.Hitch.Total	W	Total trailer hitch PowerDelta
PwrD.Hitch.Trq_x PwrD.Hitch.Trq_y PwrD.Hitch.Trq_z	PowerDelta.Hitch.Trq_x PowerDelta.Hitch.Trq_y PowerDelta.Hitch.Trq_z	W	PowerDelta due to hitch torques
PwrD.Inert_Chassis	PowerDelta.Inert_Chassis	W	PowerDelta due to inertia of the vehicle body
PwrD.PT.Battery_LV	PowerDelta.PT.Battery_LV	W	PowerDelta for the low voltage battery
PwrD.PT.Battery_HV	PowerDelta.PT.Battery_HV	W	PowerDelta for the high voltage battery
PwrD.PT.CDiff	PowerDelta.PT.CDiff	W	PowerDelta for the central differential
PwrD.PT.Clutch	PowerDelta.PT.Clutch	W	PowerDelta for the clutch
PwrD.PT.DriveLine	PowerDelta.PT.DriveLine	W	PowerDelta for the total driveline (all differentials + hang-on clutch + shafts)
PwrD.PT.Engine	PowerDelta.PT.Engine	W	PowerDelta for the engine
PwrD.PT.FDiff	PowerDelta.PT.FDiff	W	PowerDelta for the front differential
PwrD.PT.GearBox	PowerDelta.PT.GearBox	W	PowerDelta for the gearbox
PwrD.PT.GearBoxM PwrD.PT.GearBoxM<iM>	PowerDelta.PT.GearBoxM[iM]	W	PowerDelta for the electric motor gearbox iM
PwrD.PT.HangOn	PowerDelta.PT.HangOn	W	PowerDelta for the hang-on clutch
PwrD.PT.Inert	PowerDelta.PT.Inert	W	PowerDelta due to inertia of the powertrain components (engine, clutch) not passed to driveline
PwrD.PT.Inert_DL	PowerDelta.PT.Inert_DL	W	PowerDelta due to inertia of the driveline
PwrD.PT.ISG	PowerDelta.PT.ISG	W	PowerDelta for the starter generator ISG
PwrD.PT.Motor PwrD.PT.Motor<iM>	PowerDelta.PT.Motor[iM]	W	PowerDelta for the electric motor iM
PwrD.PT.PlanetGear	PowerDelta.PT.PlanetGear	W	PowerDelta for the planet gearbox (only in PowerSplit powertrain)
PwrD.PT.PowerSupply	PowerDelta.PT.PowerSupply	W	PowerDelta for the power supply
PwrD.PT.RDiff	PowerDelta.PT.RDiff	W	PowerDelta for the rear differential
PwrD.PT.Shafts	PowerDelta.PT.Shafts	W	PowerDelta for all shafts torques
PwrD.PT.Spring_DL	PowerDelta.PT.Spring_DL	W	PowerDelta due to shaft spring torque in the flexible driveline
PwrD.PT.Total	PowerDelta.PT.Total	W	Total powertrain PowerDelta
PwrD.Susp.Buffer<pos>	PowerDelta.Susp[i].Buffer	W	PowerDelta due to the buffer of suspension <pos>
PwrD.Susp.Damper<pos>	PowerDelta.Susp[i].Damper	W	PowerDelta due to damper of suspension <pos>
PwrD.Susp.Spring<pos>	PowerDelta.Susp[i].Spring	W	PowerDelta due to the spring of suspension <pos>
PwrD.Susp.Stabi<pos>	PowerDelta.Susp[i].Stabi	W	PowerDelta due to the stabi of suspension <pos>
PwrD.Susp.Total	PowerDelta.Susp_Total	W	Total suspension PowerDelta
PwrD.Susp.Total<pos>	PowerDelta.Susp[i].Total	W	Total suspension PowerDelta of suspension <pos>
PwrD.Tire.CambDefl<pos>	PowerDelta.Tire[i].CambDefl	W	PowerDelta due to camber deflection of tire <pos>

Name UAQ	Name C-Code	Unit	Info
PwrD.Tire.LatSlip<pos>	PowerDelta.Tire[i].LatSlip	W	PowerDelta due to lateral slip of tire <pos>
PwrD.Tire.LongSlip<pos>	PowerDelta.Tire[i].LongSlip	W	PowerDelta due to longitudinal slip of tire <pos>
PwrD.Tire.RollResist<pos>	PowerDelta.Tire[i].RollResist	W	PowerDelta due to rolling resistance of tire <pos>
PwrD.Tire.ToeSlip<pos>	PowerDelta.Tire[i].ToeSlip	W	PowerDelta due to toe slip of tire <pos>
PwrD.Tire.Total	PowerDelta.Tire_Total	W	Total tire PowerDelta
PwrD.Tire.Total<pos>	PowerDelta.Tire[i].Total	W	Total tire PowerDelta of tire <pos>
PwrD.Tire.VertDefl<pos>	PowerDelta.Tire[i].VertDefl	W	PowerDelta due to vertical deflection of tire <pos>
PwrD.Total	PowerDelta.Total	W	Total PowerDelta (power loss/gain)

24.13.2 PowerLoss

Name UAQ	Name C-Code	Unit	Info
PwrL.Aero	PowerLoss.Aero	W	Total power loss due to aerodynamics
PwrL.Brake	PowerLoss.Brake	W	Total power loss in the brakes
PwrL.Grvt	PowerLoss.Grvt	W	Total power loss due to road inclination
PwrL.Hitch	PowerLoss.Hitch	W	Total power loss due to trailer hitch forces
PwrL.Inert_Chassis	PowerLoss.Inert_Chassis	W	Total power loss due to inertia of vehicle body
PwrL.PT	PowerLoss.PT	W	Total power loss in the powertrain
PwrL.Susp	PowerLoss.Susp	W	Total power loss in the suspensions
PwrL.Tire	PowerLoss.Tire	W	Total power loss in the tires
PwrL.Total	PowerLoss.Total	W	Total power loss

24.13.3 PowerStore

Name UAQ	Name C-Code	Unit	Info
PwrS.Aero	PowerStore.Aero	W	Total power store due to aerodynamics
PwrS.Brake	PowerStore.Brake	W	Total power store in the brakes
PwrS.Grvt	PowerStore.Grvt	W	Total power store due to road inclination
PwrS.Hitch	PowerStore.Hitch	W	Total power store due to trailer hitch forces
PwrS.Inert_Chassis	PowerStore.Inert_Chassis	W	Total power store due to inertia of vehicle body
PwrS.PT	PowerStore.PT	W	Total power store in the powertrain
PwrS.Susp	PowerStore.Susp	W	Total power store in the suspensions
PwrS.Tire	PowerStore.Tire	W	Total power store in the tires
PwrS.Total	PowerStore.Total	W	Total power store

24.14 Sensors

24.14.1 Inertial Sensor

The quantities in the column “Simulink“ refer to those with respect to the “Car InertialSensorTrafficObject“ block in the CarMaker Simulink library.

Name UAQ	Name C-Code	Name Simulink	Frame	Unit	Info (data type, if not float)
<Nb> :=0, .., n (number)					
-	InertialSensor[Nb].SignalMask	SignalMask		-	Signal frame selection (unsigned long)
-	InertialSensor[Nb].Pos_B[0] InertialSensor[Nb].Pos_B[1] InertialSensor[Nb].Pos_B[2]	Pos_B x Pos_B y Pos_B z	FrB	m	Position of body sensor with name <nm> in FrB
-	InertialSensor[Nb].OBO_0[0] InertialSensor[Nb].OBO_0[1] InertialSensor[Nb].OBO_0[2]	OBO_0 x OBO_0 y OBO_0 z	Fr0	-	Sensor position relative to mounted body coordinate system as defined in CarMaker Vehicle GUI
-	InertialSensor[Nb].OBO_B[0] InertialSensor[Nb].OBO_B[1] InertialSensor[Nb].OBO_B[2]	OBO_B x OBO_B y OBO_B z	FrB	-	Sensor position relative to mounted body coordinate system as defined in CarMaker Vehicle GUI
Sensor.Inertial.<nm>.Acc_0.x Sensor.Inertial.<nm>.Acc_0.y Sensor.Inertial.<nm>.Acc_0.z	InertialSensor[Nb].Acc_0[0] InertialSensor[Nb].Acc_0[1] InertialSensor[Nb].Acc_0[2]	Acc_0 x Acc_0 y Acc_0 z	Fr0	m/s^2	Translational acceleration of body sensor with name <nm> in global frame
Sensor.Inertial.<nm>.Acc_B.x Sensor.Inertial.<nm>.Acc_B.y Sensor.Inertial.<nm>.Acc_B.z	InertialSensor[Nb].Acc_B[0] InertialSensor[Nb].Acc_B[1] InertialSensor[Nb].Acc_B[2]	Acc_B x Acc_B y Acc_B z	Fr1 or Fr2	m/s^2	Translational acceleration of body sensor with name <nm> in body frame (Fr1; Fr2)
Sensor.Inertial.<nm>.Alpha_0.x Sensor.Inertial.<nm>.Alpha_0.y Sensor.Inertial.<nm>.Alpha_0.z	InertialSensor[Nb].Alpha_0[0] InertialSensor[Nb].Alpha_0[1] InertialSensor[Nb].Alpha_0[2]	Alpha_0 x Alpha_0 y Alpha_0 z	Fr0	rad/s^2	Rotational acceleration of body sensor with name <nm> in global frame
Sensor.Inertial.<nm>.Alpha_B.x Sensor.Inertial.<nm>.Alpha_B.y Sensor.Inertial.<nm>.Alpha_B.z	InertialSensor[Nb].Alpha_B[0] InertialSensor[Nb].Alpha_B[1] InertialSensor[Nb].Alpha_B[2]	Alpha_B x Alpha_B y Alpha_B z	Fr1 or Fr2	rad/s^2	Rotational acceleration of body sensor with name <nm> in body frame (Fr1; Fr2)
Sensor.Inertial.<nm>.Omega_0.x Sensor.Inertial.<nm>.Omega_0.y Sensor.Inertial.<nm>.Omega_0.z	InertialSensor[Nb].Omega_0[0] InertialSensor[Nb].Omega_0[1] InertialSensor[Nb].Omega_0[2]	Omega_0 x Omega_0 y Omega_0 z	Fr0	rad/s	Rotational velocity of body sensor with name <nm> in global frame
Sensor.Inertial.<nm>.Omega_B.x Sensor.Inertial.<nm>.Omega_B.y Sensor.Inertial.<nm>.Omega_B.z	InertialSensor[Nb].Omega_B[0] InertialSensor[Nb].Omega_B[1] InertialSensor[Nb].Omega_B[2]	Omega_B x Omega_B y Omega_B z	Fr1 or Fr2	rad/s	Rotational velocity of body sensor with name <nm> in body frame (Fr1; Fr2)
Sensor.Inertial.<nm>.Pos_0.x Sensor.Inertial.<nm>.Pos_0.y Sensor.Inertial.<nm>.Pos_0.z	InertialSensor[Nb].Pos_0[0] InertialSensor[Nb].Pos_0[1] InertialSensor[Nb].Pos_0[2]	Pos_0 x Pos_0 y Pos_0 z	Fr0	m	Position of body sensor with name <nm> in global frame
Sensor.Inertial.<nm>.Vel_0.x Sensor.Inertial.<nm>.Vel_0.y Sensor.Inertial.<nm>.Vel_0.z	InertialSensor[Nb].Vel_0[0] InertialSensor[Nb].Vel_0[1] InertialSensor[Nb].Vel_0[2]	Vel_0 x Vel_0 y Vel_0 z	Fr0	m/s	Translational velocity of body sensor with name <nm> in global frame
Sensor.Inertial.<nm>.Vel_B.x Sensor.Inertial.<nm>.Vel_B.y Sensor.Inertial.<nm>.Vel_B.z	InertialSensor[Nb].Vel_B[0] InertialSensor[Nb].>Vel_B[1] InertialSensor[Nb].Vel_B[2]	Vel_B x Vel_B y Vel_B z	Fr1 or Fr2	m/s	Translational velocity of body sensor with name <nm> in body frame (Fr1; Fr2)

24.14.2 Slip Angle Sensor

Name UAQ	Name C-Code	Name Simulink	Unit	Info (data type, if not double)
<Nb> :=0, .., n (number)				
Sensor.SAngle.<nm>.Ang	SAngleSensor[Nb].Angle	-	rad	Sideslip angle

24.14.3 Object Sensor

General

The quantities in the column “Simulink“ refer to those with respect to the “Car ObjectSensor“ block in the CarMaker Simulink library.

Name UAQ	Name C-Code	Name Simulink	Unit	Info (data type, if not double)
<Nb> :=0, .., n (number) <preQ> := Sensor.Object.<name> Pointer tObjectSensor *pOS = &ObjectSensor[<SensorId>] or ObjectSensor_GetByIndex(<SensorId>) or ObjectSensor_GetByIndex(ObjectSensor_FindIndexForName(<name>))				
<preQ>.DrvLaneCurv	pOS->DrvLaneCurv	DrvLaneCurv	1/m	Predicted driving lane curvature of sensor with the name <nm>
<preQ>.rx_ext <preQ>.ry_ext <preQ>.rz_ext	pOS->rot_zyx_ext[0] pOS->rot_zyx_ext[1] pOS->rot_zyx_ext[2]	Rot_zyx x Rot_zyx y Rot_zyx z	rad	External sensor rotation with ZYX-orientation order
<preQ>.TimeStamp	pOS->TimeStamp	-	s	Time-stamp of updated sensor signals
<preQ>.tx_ext <preQ>.ty_ext <preQ>.tz_ext	pOS->.t_ext[0] pOS->t_ext[1] pOS->t_ext[2]	-	m	External sensor travel, expressed in mounted frame
Sensor.Object.nSensors	ObjectSensorCount	-	-	Number of Sensors defined (integer)

Relevant Target

Name UAQ	Name C-Code	Name Simulink	Unit	Info (data type, if not double)
<preQ> := Sensor.Object.<name>.relvTgt Pointer tObjectSensor *pOS = &ObjectSensor[<SensorId>] or ObjectSensor_GetByIndex(<SensorId>) or ObjectSensor_GetByIndex(ObjectSensor_FindIndexForName(<name>)) <preC> := pOS->relvTarget				
<preQ>.dtct	pOS->Targ_Dtct	TargetDetected	-	Flag: a relevant target is detected (boolean)
<preQ>.ObjId	pOS->relvTargetObjId	relvTargetObjId	-	Global object Id of the relevant target: - value (integer) =-1 stands for no detection - value individual to each scenario - value not contiguous, so it is unsuitable for use as an array index
<preQ>.ImgArea_LeftP <preQ>.ImgArea_NearP <preQ>.ImgArea_RightP	<preC>.ImgArea_LeftP <preC>.ImgArea_NearP <preC>.ImgArea_RightP	-	-	Projection of the target points on the sensor image area
<preQ>.IncAngle_alpha <preQ>.IncAngle_beta	<preC>.incangle[0] <preC>.incangle[1]	-	rad	Angle of incidence in the nearest point
<preQ>.RefPnt.alpha <preQ>.NearPnt.alpha	<preC>.RefPnt.alpha_p <preC>.NearPnt.alpha_p	-	rad	Bearing azimut of the relevant target (nearest point and reference point)
<preQ>.RefPnt.ds_p <preQ>.NearPnt.ds_p	<preC>.RefPnt.ds_p <preC>.NearPnt.ds_p	-	m	Distance to the relevant target (nearest point and reference point)
<preQ>.RefPnt.ds.x <preQ>.RefPnt.ds.y <preQ>.RefPnt.ds.z <preQ>.NearPnt.ds.x <preQ>.NearPnt.ds.y <preQ>.NearPnt.ds.z	<preC>.RefPnt.ds[0] <preC>.RefPnt.ds[1] <preC>.RefPnt.ds[2] <preC>.NearPnt.ds[0] <preC>.NearPnt.ds[1] <preC>.NearPnt.ds[2]	-	m	Distance to the relevant target in x,y,z-direction in sensor frame (nearest point and reference point)
<preQ>.RefPnt.dv_p <preQ>.NearPnt.dv_p	<preC>.RefPnt.dv_p <preC>.NearPnt.dv_p	-	m/s	Relative radial speed of the relevant target (nearest point and reference point)
<preQ>.RefPnt.dv.x <preQ>.RefPnt.dv.y <preQ>.RefPnt.dv.z <preQ>.NearPnt.dv.x <preQ>.NearPnt.dv.y <preQ>.NearPnt.dv.z	<preC>.RefPnt.dv[0] <preC>.RefPnt.dv[1] <preC>.RefPnt.dv[2] <preC>.NearPnt.dv[0] <preC>.NearPnt.dv[1] <preC>.NearPnt.dv[2	-	m/s	Speed of the relevant target in x,y,z-direction in sensor frame (nearest point and reference point)
<preQ>.RefPnt.r_zyx.x <preQ>.RefPnt.r_zyx.y <preQ>.RefPnt.r_zyx.z	<preC>.RefPnt.r_zyx[0] <preC>.RefPnt.r_zyx[1] <preC>.RefPnt.r_zyx[2]	-	rad	Orientation of relevant target in the reference point referring to the sensor frame with rotation order z-y-x
<preQ>.RefPnt.theta <preQ>.NearPnt.theta	<preC>.RefPnt.theta_p <preC>.NearPnt.theta_p	-	rad	Bearing elevation of the relevant target (nearest point and reference point)

Object List

The quantities in the column “Simulink“ refer to those with respect to the “Car ObjectSensor Object“ block in the CarMaker Simulink library.

Name UAQ	Name C-Code	Name Simulink	Unit	Info (data type, if not double)
<Nb> :=0, .., n (number) <preQ> := Sensor.Object.<name_sensor>.Obj.<name_object> Pointer tObjectSensorObj *pOSO = ObjectSensor_GetObject(<Sensor Id>, <Internal TrafficId>) or ObjectSensor_GetObjectByObjId(<Sensor Id>, <Global ObjId>) or ObjectSensor_GetObjectForName(<Sensor Name>, <Traffic Object Name>)				
-		TrafficObjectId	-	Traffic (internal) object’s identification number (integer)
-	pOSO->ObjId	GlobalObjectId	-	Global object’s identification number (integer)
-	pOSO->h pOSO->l pOSO->w	height length width	m	Traffic object’s height, length, width
-	pOSO->zOff	zOff	m	Distance between the traffic object’s bottom and the road
<preQ>.dtct	pOSO->dtct	dtct	-	Flag: Traffic object is detected bz sensor <nm> (boolean)
<preQ>.ImgArea_LeftP <preQ>.ImgArea_NearP <preQ>.ImgArea_RightP	pOSO->ImgArea_LeftP pOSO->ImgArea_NearP pOSO->ImgArea_RightP	ImgArea_LeftP ImgArea_NearP ImgArea_RightP	-	Projection of the object points on the sensor image area
<preQ>.IncAngle_alpha <preQ>.IncAngle_beta	pOSO->incangle[0] pOSO->incangle[1]	IncAngle 0 IncAngle 1	rad	Angle of incidence in the nearest point of the traffic object
<preQ>.InLane	pOSO->InLane	-	-	Flag: traffic object is within the vehicle driving lane (only for ’LaneTracking’ detection) (boolean)
<preQ>.obsv	pOSO->obsv	obsv	-	Flag: traffic object is inside the observation area of sensor <nm> (all sensor quantities are calculated) (boolean)
<preQ>.RefPnt.alpha <preQ>.NearPnt.alpha	pOSO->RefPnt.alpha_p pOSO->NearPnt.alpha_p	RefPnt alpha_p NearPnt alpha_p	rad	Bearing azimut of the traffic object (nearest point and reference point)
<preQ>.RefPnt.ds_p <preQ>.NearPnt.ds_p	pOSO->RefPnt.ds_p pOSO->NearPnt.ds_p	RefPnt ds_p NearPnt ds_p	m	Distance from sensor to traffic object (nearest point and reference point)
<preQ>.RefPnt.ds.x <preQ>.RefPnt.ds.y <preQ>.RefPnt.ds.z <preQ>.NearPnt.ds.x <preQ>.NearPnt.ds.y <preQ>.NearPnt.ds.z	pOSO->RefPnt.ds[0] pOSO->RefPnt.ds[1] pOSO->RefPnt.ds[2] pOSO->NearPnt.ds[0] pOSO->NearPnt.ds[1] pOSO->NearPnt.ds[2]	RefPnt ds x RefPnt ds y RefPnt ds z NearPnt ds x NearPnt ds y NearPnt ds z	m	Distance from sensor to traffic object in x,y,z-direction in sensor frame (nearest point and reference point)
<preQ>.RefPnt.dv_p <preQ>.NearPnt.dv_p	pOSO->RefPnt.dv_p pOSO->NearPnt.dv_p	RefPnt dv_p NearPnt dv_p	m/s	Relative radial speed (distance change rate) (nearest point and reference point)
<preQ>.RefPnt.dv.x <preQ>.RefPnt.dv.y <preQ>.RefPnt.dv.z <preQ>.NearPnt.dv.x <preQ>.NearPnt.dv.y <preQ>.NearPnt.dv.z	pOSO->RefPnt.dv[0] pOSO->RefPnt.dv[1] pOSO->RefPnt.dv[2] pOSO->NearPnt.dv[0] pOSO->NearPnt.dv[1] pOSO->NearPnt.dv[2]	RefPnt dv x RefPnt dv y RefPnt dv z NearPnt dv x NearPnt dv y NearPnt dv z	m/s	Speed of the traffic object in x,y,z-direction in sensor frame (nearest point and reference point)
<preQ>.RefPnt.r_zyx.x <preQ>.RefPnt.r_zyx.y <preQ>.RefPnt.r_zyx.z	pOSO->RefPnt.rzyx[0] pOSO->RefPnt.rzyx[1] pOSO->RefPnt.rzyx[2]	RefPnt r_zyx x RefPnt r_zyx y RefPnt r_zyx z NearPnt r_zyx x NearPnt r_zyx y NearPnt r_zyx z	rad	Orientation of traffic object <nm_O> in the reference point referring to the sensor frame with rotation order z-y-x
<preQ>.RefPnt.theta <preQ>.NearPnt.theta	pOSO->RefPnt.theta_p pOSO->NearPnt.theta_p	RefPnt theta_p NearPnt theta_p	rad	Bearing elevation of the traffic object (nearest point and reference point)

24.14.4 Free Space Sensor

General

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<Nb> :=0, .., n (number of sensors) <preQ> := Sensor.FSpace.<name> <preC> := FSpaceSensor[Nb]			

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<preQ>.rx_ext <preQ>.ry_ext <preQ>.rz_ext	<preC>.rot_zyx_ext[0] <preC>.rot_zyx_ext[1] <preC>.rot_zyx_ext[2]	rad	Additional external rotation of sensor, expressed in mounted frame with ZYX-orientation order
<preQ>.TimeStamp	<preC>.TimeStamp	s	Time-stamp of updated sensor signal
<preQ>.tx_ext <preQ>.ty_ext <preQ>.tz_ext	<preC>.t_ext[0] <preC>.t_ext[1] <preC>.t_ext[2]	m	Additional external travel of sensor, expressed in mounted frame

Segments

The segment numeration begins on the upper left and ends on the lower right.

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<Nb> :=0, .., n (number of sensors) <Nb_S> :=0, .., nS (number of segments) <preQ> := Sensor.FSpace.<name> <preC> := FSpaceSensor[Nb]			
<preQ>.Segm.<Nb_S>.alpha	<preC>.Segm[Nb_S].alpha_p	rad	Bearing azimut of the nearest point in sensor frame (polar coordinates)
<preQ>.Segm.<Nb_S>.ds	<preC>.Segm[Nb_S].ds_p	m	Distance to the nearest point
<preQ>.Segm.<Nb_S>.ds.x <preQ>.Segm.<Nb_S>.ds.y <preQ>.Segm.<Nb_S>.ds.z	<preC>.Segm[Nb_S].ds[0] <preC>.Segm[Nb_S].ds[1] <preC>.Segm[Nb_S].ds[2]	m	Position of the nearest point in x,y,z-coordinates of sensor frame Not provided by FSpace Sensor Plus
<preQ>.Segm.<Nb_S>.dv	<preC>.Segm[Nb_S].dv_p	m/s	Relative radial speed (distance change rate) of the nearest point
<preQ>.Segm.<Nb_S>.dv.x <preQ>.Segm.<Nb_S>.dv.y <preQ>.Segm.<Nb_S>.dv.z	<preC>.Segm[Nb_S].dv[0] <preC>.Segm[Nb_S].dv[1] <preC>.Segm[Nb_S].dv[2]	m/s	Velocity of the nearest n x,y,z-coordinates of sensor frame Not provided by FSpace Sensor Plus
<preQ>.Segm.<Nb_S>.theta	<preC>.Segm[Nb_S].theta_p	rad	Bearing elevation of the nearest point in sensor frame (polar coordinates)
<preQ>.Segm.<Nb_S>.ObjId	<preC>.Segm[Nb_S].ObjId	-	Identification number of detected global object:: - value (integer) =-1 means no detection, free space - value individual to each scenario - value not contiguous, so it is unsuitable for use as an array index
Additional quantities for FSpace Sensor Plus			
<preQ>.Segm.<Nb_S>.nuv.x <preQ>.Segm.<Nb_S>.nuv.y <preQ>.Segm.<Nb_S>.nuv.z	<preC>.Segm[Nb_S].nuv[0] <preC>.Segm[Nb_S].nuv[1] <preC>.Segm[Nb_S].nuv[2]	Fr0	Normal vector of the detected point
<preQ>.Segm.<Nb_S>.MatId	<preC>.Segm[Nb_S].MatId		Material Id
<preQ>.Segm.<Nb_S>.CosIncid	<preC>.Segm[Nb_S].CosIncid		Cosine of angle between sensor ray and the object's surface normal at hit point

24.14.5 Traffic Sign Sensor

General

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<Nb> :=0, .., n (number)			
Sensor.TSign.<nm>.nSign	TSignSensor[Nb].nSign	-	Total number of detected signs (integer)
Sensor.TSign.<nm>.rx_ext Sensor.TSign.<nm>.ry_ext Sensor.TSign.<nm>.rz_ext	TSignSensor[Nb].rot_zyx_ext[0] TSignSensor[Nb].rot_zyx_ext[1] TSignSensor[Nb].rot_zyx_ext[2]	rad	Additional external rotation of sensor, expressed in mounted frame with rotation sequence ZYX
Sensor.TSign.<nm>.TimeStamp	TSignSensor[Nb].TimeStamp	s	Time stamp of updated sensor signals
Sensor.TSign.<nm>.tx_ext Sensor.TSign.<nm>.ty_ext Sensor.TSign.<nm>.tz_ext	TSignSensor[Nb].t_ext[0] TSignSensor[Nb].t_ext[1] TSignSensor[Nb].t_ext[2]	m	Additional external travel of sensor, expressed in mounted frame

Detected signs

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
if the sensor selects all: <pre> := Sensor.TSign.<sensorname>.<i> with i=0...39 if the sensor selects specified signs: <pre> := Sensor.TSign.<sensorname>.<sign>.<i> with i=0...3			
<pre>.ds	TSignSensor[Nb].Sign[Nb_TS].ds_p	m	Distance to traffic sign position (polar coordinates)

<pre>.ds.x
<pre>.ds.y
<pre>.ds.z
	TSignSensor[Nb].Sign[Nb_TS].ds[0]
	TSignSensor[Nb].Sign[Nb_TS].ds[1]
	TSignSensor[Nb].Sign[Nb_TS].ds[2]
		m	Distance to traffic sign position in sensor frame or mounted frame

<pre>.Main.Id.	TSignSensor[Nb].Sign[Nb_TS].main.signId	-	Traffic sign identification given by road (integer)

<pre>.Main.val0
<pre>.Main.val1
	TSignSensor[Nb].Sign[Nb_TS].main.val[0]
	TSignSensor[Nb].Sign[Nb_TS].main.val[1]
		-	User defined sign attribute values (float)

24.14.6 Line Sensor

The UAQ LineDetect.* are meant for visualization purposes only.

General

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<Nb> :=0, .., n (number)			
Sensor.Line.<nm>.nLine_Left Sensor.Line.<nm>.nLine_Right	LineSensor[Nb].LLines.nLine LineSensor[Nb].RLines.nLine	-	Number of lines detected at left Number of lines detected at right (integer)
Sensor.Line.<nm>.rx_ext Sensor.Line.<nm>.ry_ext Sensor.Line.<nm>.rz_ext	LineSensor[Nb].rot_zyx_ext[0] LineSensor[Nb].rot_zyx_ext[1] LineSensor[Nb].rot_zyx_ext[2]	rad	Additional external rotation of sensor, expressed in mounted frame with ZYX-orientation order
Sensor.Line.<nm>.TimeStamp	LineSensor[Nb].TimeStamp	s	Time stamp of updated sensor signal
Sensor.Line.<nm>.tx_ext Sensor.Line.<nm>.ty_ext Sensor.Line.<nm>.tz_ext	LineSensor[Nb].t_ext[0] LineSensor[Nb].t_ext[1] LineSensor[Nb].t_ext[2]	m	Additional external travel of sensor, expressed in mounted frame

Detected lines

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<pre> := Sensor.Line.<sensorname>.LLines.<i> or Sensor.Line.<sensorname>.RLines.<i> <pre2> := LineSensor[<sensornumber>].LLines.L[<i>] or LineSensor[<sensornumber>].RLines.L[<i>] with i=1...10 (1 is the most nearest line) for <pre> and i=0...9 for <pre2>			
<pre>.ColorCode	<pre2>.colorCode	-	User defined color index of detected line left/right (only for sensor, independent of color displayed in IPGMovie) (integer)
<pre>.Height	<pre2>.lineHeight	m	Line height used for traffic barrier
<pre>.Type	<pre2>.type	-	Line type (integer): 1 = continuous single line 2 = broken line 3 = dotted line 4 = double line, continuous 5 = double line, broken 6 = double line, dotted 7 = double line, continuous right, broken left 8 = double line, continuous left, broken right 9 = double line, continuous right, dotted left 10 = double line, continuous left, dotted right 12 = Traffic barrier
<pre>.Width	<pre2>.lineWidth	m	Line width
<pre>.Id	<pre2>.id	-	Line identification number (integer)

24.14.7 Road Sensor

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<preQ> := Sensor.Road.<name> <preC> := RoadSensor[i]			
<preQ>.Lane.Act.isRight <preQ>.Lane..OnLeft.isRight <preQ>.Lane..OnRight.isRight	<preC>.Act.isRight <preC>.OnLeft.isRight <preC>.OnRight.isRight	-	Indicates if the actual lane, at which the preview point along the path is, is on right side (same for lanes on left / right side) in path direction
<preQ>.Lane.Act.LaneId <preQ>.Lane.OnLeft.LaneId <preQ>.Lane.OnRight.LaneId	<preC>.Act.LaneId <preC>.OnLeft.LaneId <preC>.OnRight.LaneId	-	Indicates the lane Id at which the preview point along the path is, same for lanes on left / right side (-1 means no lane found)
<preQ>.Lane.Act.tMidLane <preQ>.Lane.OnLeft.tMidLane <preQ>.Lane.OnRight.tMidLane	<preC>.Act.tMidLane <preC>.OnLeft.tMidLane <preC>.OnRight.tMidLane	m	Lateral offset of actual lane mid to the route center line (same for lanes on left / right side) at preview point along path
<preQ>.Lane.Act.Type <preQ>.Lane.OnLeft.Type <preQ>.Lane.OnRight.Type	<preC>.Act.Type <preC>.OnLeft.Type <preC>.OnRight.Type	-	Type of the actual lane at preview point along path (same for lanes on left / right side): 0: legally driveable road 1: bicycle lane 2: lane only for pedestrian 3: traffic island 4: border lane 5: roadside
<preQ>.Lane.Act.Width <preQ>.Lane.OnLeft.Width <preQ>.Lane.OnRight.Width	<preC>.Act.Width <preC>.OnLeft.Width <preC>.OnRight.Width	m	Width of the actual roadway lane and the lanes on left / right side at preview point along the path
<preQ>.Lane.nLeft <preQ>.Lane.nRight	<preC>.Lanes.nLanesL <preC>.Lanes.nLanesR	-	Number of left, right roadway lanes along path / route (integer)
<preQ>.onRoad	<preC>.onRoad	-	Flag: preview point is on the road (boolean) 0 = not on the road, RoadSensor quantities are not updated 1 = on the road, RoadSensor quantities are updated
<preQ>.PreviewDist	<preC>.PreviewDist	m	Sensor preview distance. Can be changed via DVA or by modifying the C-variable.
<preQ>.RMarker.Attrib.<j>	<preC>.MarkerAttirb[j]	-	User defined attribute <j> of detected road marker (float)
<preQ>.Path.CurveXY <preQ>.Route.CurveXY	<preC>.Path.CurveXY <preC>.Route.CurveXY	1/m	Route / path curvature at x-y plane at preview point
<preQ>.Path.DevAng <preQ>.Route.DevAng	<preC>.Path.Deviation.Ang <preC>.Route.Deviation.Ang	rad	Deviation angle at preview point along path / route
<preQ>.Path.DevDist <preQ>.Route.DevDist	<preC>.Path.Deviation.Dist <preC>.Route.Deviation.Dist	m	Deviation distance at preview point along path projected to the sensor point; deviation distance to the route
<preQ>.Path.LatSlope <preQ>.Route.LatSlope	<preC>.Path.LatSlope <preC>.Route.LatSlope	rad	Lateral road slope at preview point along path / route
<preQ>.Path.LongSlope <preQ>.Route.LongSlope	<preC>.Path.LongSlope <preC>.Route.LongSlope	rad	Longitudinal road slope at preview point along path / route
<preQ>.Path.tx <preQ>.Path.ty <preQ>.Path.tz <preQ>.Route.tx <preQ>.Route.ty <preQ>.Route.tz	<preC>.Path.P_0[0] <preC>.Path.P_0[1] <preC>.Path.P_0[2] <preC>.Route.P_0[0] <preC>.Route.P_0[1] <preC>.Route.P_0[2]	m	Global position (Fr0) of the preview point along the route and path
<preQ>.tx <preQ>.ty <preQ>.tz	<preC>.P_0[0] <preC>.P_0[1] <preC>.P_0[2]	m	Global position (Fr0) of the preview point along the vehicle direction
<preQ>.Z_0.x <preQ>.Z_0.y <preQ>.Z_0.z	<preC>.Z_0[0] <preC>.Z_0[1] <preC>.Z_0[2]	-	Road normal vector at preview point along the vehicle direction

24.14.8 Collision Sensor

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<pre> := Sensor.Collision			
<pre>.Vhcl.Fr1.Count		-	Counter for detected collision with the vehicle body Fr1
<pre>.Vhcl.Fr1.ObjId		-	Detected collision of vehicle body Fr1 with the global object ObjId: - value (integer) =-1 stands for no detection - value individual to each scenario - value not contiguous, so it is unsuitable for use as an array index
<pre>.Vhcl.W<pos>.Count		-	Counter for detected collision with the vehicle wheel <pos>
<pre>.Vhcl.W<pos>.ObjId		-	Detected collision of vehicle wheel <pos> with the global object ObjId: - value (integer) =-1 stands for no detection - value individual to each scenario - value not contiguous, so it is unsuitable for use as an array index

24.14.9 Radar Sensor

General

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<NoS>:= 0,... , n (sensors) <preQ>:= Sensor.Radar <preC>:= RadarSensor[NoS].GlobalInf[0]			
<preQ>.nSensors	RadarCount	-	number of sensors
<preQ>.<name>.RolCount	<preC>.RolCount	-	rolling cycle counter (integer)
<preQ>.<name>.nObj	<preC>.nObj	-	number of detected objects (integer)
<preQ>.<name>.RelvTgt	<preC>.RelvTgt	-	number of the relevant target in the object list (integer)
<preQ>.<name>.nLanesL <preQ>.<name>.nLanesR	<preC>.nLanesL <preC>.nLanesR	-	number of lanes left/right (integer)
<preQ>.<name>.DistToLeftBorder <preQ>.<name>.DistToRightBorder	<preC>.DistToLeftBorder <preC>.DistToRightBorder	m	distance to left/right border
<preQ>.<name>.TransmitPower	RadarSensor[NoS].TransmitPower	[dBm]	(DVA) use this quantity to alter the transmit power during simulation

Object List

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<NoO:= 0,... , m (objects) <preQ>:= Sensor.Radar.<name>.Obj<NoO> <preC>:= RadarSensor[NoS].ObjList[NoO]			
<preQ>.ObjId	<preC>.ObjId	-	global ID to identify object
<preQ>.LeftInGroup	<preC>.LeftInGroup	-	object id of left traffic object in group, if sensor sees two or more traffic objects as one (integer)
<preQ>.MeasStat	<preC>.MeasStat	-	measurement status (integer) 0: no object 1: new object 2: (currently not used) 3: object measured
<preQ>.Dist <preQ>.DistX <preQ>.DistY <preQ>.DistZ <preQ>.RelCourseAngle	<preC>.Dist <preC>.DistX <preC>.DistY <preC>.DistZ <preC>.RelCourseAngle	m m m m rad	distance from sensor to “reference” point of the detected object longitudinal displacement in sensor frame lateral displacement in sensor frame vertical displacement in sensor frame relative course angle in sensor frame
<preQ>.Vrel <preQ>.VrelX <preQ>.VrelY <preQ>.ArelX	<preC>.Vrel <preC>.VrelX <preC>.VrelY <preC>.ArelX	m/s m/s m/s m/s^2	velocity of “reference” point of the detected object relative to sensor relative longitudinal velocity relative lateral velocity relative longitudinal acceleration

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<preQ>.Length <preQ>.Width	<preC>.Length <preC>.Width	m m	length of bounding box width of bounding box
<preQ>.LengthClass <preQ>.WidthClass	<preC>.LengthClass <preC>.WidthClass	- -	length class from 0 to 7 (integer) width class from 0 to 7 (integer) 0:unknown 1: < 0.5m 2: < 1m 3: < 2m 4: < 3m 5: < 4m 6: < 6m 7: exceeds
<preQ>.RCS <preQ>.SignalStrength <preQ>.SNR	<preC>.RCS <preC>.SignalStrength <preC>.SNR	dBm^2 dBW dB	RCS - value signal signal-to-noise ratio
<preQ>.DynProp	<preC>.DynProp	-	dynamic property (integer) 0: not classified 1: stationary 2: stopped 3: moving 4: oncoming
<preQ>.ProbDetect	<preC>.ProbDetect	-	probability of detection
<preQ>.ProbExist	<preC>.ProbExist	-	probability of existence from 0 to 7 (integer) 0: not detected 1: ~ 25% 2: ~ 50% 3: ~ 75% 4: ~ 90% 5: ~ 99% 6: ~ 99.9% 7: ~ 100%
<preQ>.ProbObst	<preC>.ProbObst	-	obstacle probability

24.14.10 Global Navigation Sensor

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<preQ> := Sensor.GNav <preQ_Rec> := Sensor.GNav.Receiver <preQ_Sat> := Sensor.GNav.Sat <preQ_Rinex> := Sensor.GNav.RinexData <preQ_RinexEph> := Sensor.GNav.RinexData.Sat_Ephemeris <preC> := GNavSensor <preC_Rec> := GNavSensor.Receiver			
<preQ_Rec>.Position.ECEF.x <preQ_Rec>.Position.ECEF.y <preQ_Rec>.Position.ECEF.z	<preC_Rec>.UserPosEcefTsa[0] <preC_Rec>.UserPosEcefTsa[1] <preC_Rec>.UserPosEcefTsa[2]	m	Exact receiver position in Earth Centered Earth Fixed (ECEF) coordinate system.
<preQ_Rec>.Position.ECEF_fromPsr.x <preQ_Rec>.Position.ECEF_fromPsr.y <preQ_Rec>.Position.ECEF_fromPsr.z	<preC_Rec>.UserPosEcefTsa_fromPsr[0] <preC_Rec>.UserPosEcefTsa_fromPsr[1] <preC_Rec>.UserPosEcefTsa_fromPsr[2]	m	Receiver position in Earth Centered Earth Fixed (ECEF) coordinate system calculated from pseudorange.
<preQ_Rec>.Position.ECEF_Delta.x <preQ_Rec>.Position.ECEF_Delta.y <preQ_Rec>.Position.ECEF_Delta.z	<preC_Rec>.Delta_UserPosEcefTsa[0] <preC_Rec>.Delta_UserPosEcefTsa[1] <preC_Rec>.Delta_UserPosEcefTsa[2]	m	Difference of receiver position from pseudorange to exact position in Earth Centered Earth Fixed (ECEF) coordinate system.
<preQ_Rec>.Position.ECEFVel.x <preQ_Rec>.Position.ECEFVel.y <preQ_Rec>.Position.ECEFVel.z	<preC_Rec>.UserVelEcefTsa[0] <preC_Rec>.UserVelEcefTsa[1] <preC_Rec>.UserVelEcefTsa[2]	m/s	Velocity of the receiver in Earth Centered Earth Fixed (ECEF) coordinate system.
<preQ_Rec>.Position.LLH.Lat <preQ_Rec>.Position.LLH.Long <preQ_Rec>.Position.LLH.H	<preC_Rec>.UserPosLlhTsa[0] <preC_Rec>.UserPosLlhTsa[1] <preC_Rec>.UserPosLlhTsa[2]	rad rad m	Exact receiver position in LLH (latitudel, longitude, height) coordinate system.
<preQ_Rec>.Position.LLH_fromPsr.Lat <preQ_Rec>.Position.LLH_fromPsr.Long <preQ_Rec>.Position.LLH_fromPsr.H	<preC_Rec>.UserPosLlhTsa_fromPsr[0] <preC_Rec>.UserPosLlhTsa_fromPsr[1] <preC_Rec>.UserPosLlhTsa_fromPsr[2]	rad rad m	Receiver position in LLH (latitudel, longitude, height) coordinate system calculated from pseudorange.
<preQ_Rec>.Position.LLH_Delta.Lat <preQ_Rec>.Position.LLH_Delta.Long <preQ_Rec>.Position.LLH_Delta.H	<preC_Rec>.UserPosLlhTsa_fromPsr[0] <preC_Rec>.UserPosLlhTsa_fromPsr[1] <preC_Rec>.UserPosLlhTsa_fromPsr[2]	rad rad m	Difference of receiver position from pseudorange to exact position in LLH (latitudel, longitude, height) coordinate system.
<preQ_Rec>.Time.GPSWeek	<preC_Rec>.Time_GpsWeek	-	Number of GPS week.
<preQ_Rec>.Time.secsOfWeek	<preC_Rec>.Time_secsOfWeek	s	Secs of GPS week.
<preQ_Rec>.Time.yday	<preC_Rec>.Time_yday	-	Number of day in the year.
<preQ_Rec>.Time.ClockError	<preC_Rec>.RecClockError	m	Exact receiver clock error.
<preQ_Rec>.Time.ClockError_fromPsr	<preC_Rec>.RecClockError_fromPsr	m	Receiver clock error calculated from pseudorange.
<preQ_Rec>.Time.ClockError_Delta	<preC_Rec>.Delta_RecClockError	m	Difference of receiver clock error from pseudorange to exact receiver clock error.
<preQ_Rec>.SatNoOverElevMask	<preC_Rec>.NoVisibleSat	-	Number of satellites that are above the elevation mask.
<preQ_Rec>.SatNoDirectView	<preC_Rec>.NoVisibleSatDirect	-	Number of satellites that have a clear line of sight to the receiver.
<preQ_Rec>.GDOP	<preC_Rec>.GDOP	-	Geometrical dilution of precision.
<preQ_Rec>.TDOP	<preC_Rec>.TDOP	-	Time dilution of precision.
<preQ_Rec>.PDOP	<preC_Rec>.PDOP	-	Positional dilution of precision.
<preQ_Rec>.channel<i>.PRN	<preC_Rec>.Channel[i].PRN	-	Pseudo random noise (PRN) number of satellite visible for receiver channel i.
<preQ_Rec>.channel<i>.PseudoRange	<preC_Rec>.Channel[i].psr	m	Pseudorange of satellite visible for receiver channel i.
<preQ_Rec>.channel<i>.RangeRate	<preC_Rec>.Channel[i].rr	m/s	Range rate of satellite visible for receiver channel i.
<preQ_Rec>.channel<i>.elev	<preC_Rec>.Channel[i].elev	deg	Elevation angle of satellite visible for receiver channel i.
<preQ_Rec>.channel<i>.azim	<preC_Rec>.Channel[i].azim	deg	Azimuth angle of satellite visible for receiver channel i.
<preQ_Rec>.channel<i>.geometricRange	<preC_Rec>.Channel[i].dist	m	Geometrical distance to satellite visible for rec. channel i.
<preQ_Rec>.channel<i>.ephError	<preC_Rec>.Channel[i].ephError	m	Ephemeris error of satellite visible for receiver channel i.
<preQ_Rec>.channel<i>.ionoError	<preC_Rec>.Channel[i].ionoError	m	Ionospheric error of satellite visible for receiver channel i.
<preQ_Rec>.channel<i>.tropoError	<preC_Rec>.Channel[i].tropoError	m	Tropospheric error of satellite visible for receiver channel i.
<preQ_Rec>.channel<i>.SatelliteClockOffset	<preC_Rec>.Channel[i].dtsv	s	Satellite clock offset of satellite visible for rec. channel i.
<preQ_Sat>.sat<i>.PosECEF.x <preQ_Sat>.sat<i>.PosECEF.y <preQ_Sat>.sat<i>.PosECEF.z	<preC>.sat[i].PosECEF[0] <preC>.sat[i].PosECEF[1] <preC>.sat[i].PosECEF[2]	m	Position of satellite i in Earth Centered Earth Fixed (ECEF) coordinate system.
<preQ_Sat>.sat<i>.VelECEF.x <preQ_Sat>.sat<i>.VelECEF.y <preQ_Sat>.sat<i>.VelECEF.z	<preC>.sat[i].VelECEF[0] <preC>.sat[i].VelECEF[1] <preC>.sat[i].VelECEF[2]	m/s	Velocity of satellite i in Earth Centered Earth Fixed (ECEF) coordinate system.
<preQ_Sat>.sat<i>.elev	<preC>.sat[i].elev	deg	Elevation angle of satellite i.
<preQ_Sat>.sat<i>.azim	<preC>.sat[i].azim	deg	Azimuth angle of satellite i.

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<preQ_Sat>.sat<i>.isVisible	<preC>.sat[i].satIsVisible	-	Flag: Satellite i is over elevation mask (boolean).
<preQ_Sat>.sat<i>.DirectView	<preC>.sat[i].satDirectVisible	-	Flag: Satellite i has clear line of sight to receiver (boolean).
<preQ_Rinex>.Klobuchar.Alpha1 <preQ_Rinex>.Klobuchar.Alpha2 <preQ_Rinex>.Klobuchar.Alpha3 <preQ_Rinex>.Klobuchar.Alpha4	-	-	Ionosphere parameters A0-A3 of almanac
<preQ_Rinex>.Klobuchar.Beta1 <preQ_Rinex>.Klobuchar.Beta2 <preQ_Rinex>.Klobuchar.Beta3 <preQ_Rinex>.Klobuchar.Beta4	-	-	Ionosphere parameters B0-B3 of almanac
<preQ_RinexEph>.sat<i>.toe_GPSWeek	-	-	GPS week number - Time of Ephemeris
<preQ_RinexEph>.sat<i>.toe_secsOfWeek	-	sec	Seconds of GPS week - Time of Ephemeris
<preQ_RinexEph>.sat<i>.toc_GPSWeek	-	-	GPS week number - Time of Clock
<preQ_RinexEph>.sat<i>.toc_secsOfWeek	-	sec	Seconds of GPS week - Time of Clock
<preQ_RinexEph>.sat<i>.clock_a0 <preQ_RinexEph>.sat<i>.clock_a1 <preQ_RinexEph>.sat<i>.clock_a2	-	sec s/s s/s^2	Satellite SV clock error bias Satellite SV clock error drift Satellite SV clock error drift rate
<preQ_RinexEph>.sat<i>.C_ic	-	rad	Cosine correction of inclination angel
<preQ_RinexEph>.sat<i>.C_is	-	rad	Sine correction of inclinatin angle
<preQ_RinexEph>.sat<i>.C_uc	-	rad	Cosine correction of longitude
<preQ_RinexEph>.sat<i>.C_us	-	rad	Sine correction of longitude
<preQ_RinexEph>.sat<i>.C_rc	-	m	Cosine correction of radius of orbit
<preQ_RinexEph>.sat<i>.C_rc	-	m	Sine correction of radius of orbit
<preQ_RinexEph>.sat<i>.delta_n	-	rad/s	Mean motion difference from computed value
<preQ_RinexEph>.sat<i>.e	-	-	Eccentricity
<preQ_RinexEph>.sat<i>.i_0	-	rad	Inclination angle at Time of Ephemeris
<preQ_RinexEph>.sat<i>.iDot	-	rad/s	Rate of inclinatino angle
<preQ_RinexEph>.sat<i>.M_0	-	rad	Median anomaly at Time of Ephemeris
<preQ_RinexEph>.sat<i>.omega	-	rad	Argmument of perigee
<preQ_RinexEph>.sat<i>.omega_0	-	rad	Longitude of ascending node of orbit plane at weekly epoch
<preQ_RinexEph>.sat<i>.omegaDot	-	rad/s	Rate of right ascension
<preQ_RinexEph>.sat<i>.sqrtA	-	sqrt(m)	Square root of semi-major axis

24.14.11 Ultrasonic RSI

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<NoT> := 0..n (transducers) <NoE> := 0..m (echoes) <preQ> := Sensor.USonicRSI.<name> <preC> := USonicRSI[NoT] <preQE> := Sensor.USonicRSI.<name>.Echo.<NoE> <preCE> := USonicRSI[NoT].Echo[NoE]			
<preQ>.nEchos	<preC>.nEchos	-	number of echoes (int)
<preQE>.TimeOF	<preCE>.TimeOF	s	time of flight
<preQE>.LengthOF	<preCE>.LengthOF	m	length of flight
<preQE>.SPA	<preCE>.SPA	Pa	sound pressure amplitude
<preQE>.SPL	<preCE>.SPL	dB//1muPa	sound pressure level
<preQE>.nRefl	<preCE>.nRefl	-	number of reflections
<preQE>.Tx	<preCE>.Tx	-	transmitter to identify cross echo (int)

24.14.12 Radar RSI

Name UAQ	Name C-Code	Unit	Info (data type, if not double)
<NoT> := 0..n (transceivers) <NoC> := 0..m (channels) <preQ> := Sensor.RadarRSI.<name> <preC> := RadarRSI[NoT] <preQC> := Sensor.RadarRSI.<name>.Chan.<NoC> <preCC> := RadarRSI[NoT].Chan[NoC]			
<preQ>.nChans	<preC>.nChans	-	number of channels (int)
<preQC>.TimeOF	<preCC>.TimeOF	s	time of flight
<preQC>.LengthOF	<preCC>.LengthOF	m	length of flight
<preQC>.e_hh	<preCC>.e[0][0]	-	relative electric field amplitude, polarization: horizontally transmitted, horizontally received
<preQC>.e_hv	<preCC>.e[0][1]	-	relative electric field amplitude, polarization: horizontally transmitted, vertically received
<preQC>.e_vh	<preCC>.e[1][0]	-	relative electric field amplitude, polarization: vertically transmitted, horizontally received
<preQC>.e_vv	<preCC>.e[1][1]	-	relative electric field amplitude, polarization: vertically transmitted, vertically received
<preQC>.Dir_Tx_azi	<preCC>.Dir[0][0]	rad	direction of transmitted wave, azimuth
<preQC>.Dir_Tx_ele	<preCC>.Dir[0][1]	rad	direction of transmitted wave, elevation
<preQC>.Dir_Rx_azi	<preCC>.Dir[1][0]	rad	direction of received wave, azimuth
<preQC>.Dir_Rx_ele	<preCC>.Dir[1][1]	rad	direction of received wave, elevation
<preQC>.dshift	<preCC>.dshift	-	relative doppler shift
<preQC>.nRefl	<preCC>.nRefl	-	number of reflections
<preQC>.Tx	<preCC>.Tx	-	transmitter ID

24.15 Trailer

24.15.1 General

Name UAQ	Name C-Code	Frame	Unit	Info (data type, if not double)
<pos> := FL, FR, RL, RR <part> := Buf, Damp, Spring, Stabi <i> := 0, 1 Twin := used by the quantities for a twin wheel (describes the inner tire)				
Tr.Aero.Frc_1.x Tr.Aero.Frc_1.y Tr.Aero.Frc_1.z		Fr1	N	Aerodynamic force acting on trailer
Tr.Aero.tau_1		Fr1	rad	Angle of incidence
Tr.Aero.tau2_1		Fr1	rad	Angle of incidence (shifted by 2pi)
Tr.Aero.Trq_1.x Tr.Aero.Trq_1.y Tr.Aero.Trq_1.z		Fr1	Nm	Aerodynamic torque acting on trailer
Tr.Aero.vres_1.x Tr.Aero.vres_1.y Tr.Aero.vres_1.z		Fr1	m/s	Relative wind velocity
Tr.alHori	Trailer.ConBdy1.alHori		m/s^2	Horizontal lateral acceleration
Tr.atHori	Trailer.ConBdy1.atHori		m/s^2	Horizontal tangential acceleration
Tr.ax Tr.ay Tr.az	Trailer.ConBdy1.a_0[0] Trailer.ConBdy1.a_0[1] Trailer.ConBdy1.a_0[2]	Fr1	m/s^2	Trailer acceleration
Tr.Brake.Trq_<pos>_ext Tr.Brake.Trq_<pos>_tot Tr.Brake.Trq_PB_<pos> Tr.Brake.Trq_Reg_trg_<pos> Tr.Brake.Trq_WB_<pos>	TrBrake.Trq_ext[i] TrBrake.Trq_tot[i] TrBrake.IF.Trq_PB[i] TrBrake.IF.Trq_Reg_trg[i] TrBrake.IF.Trq_WB[i]		Nm	Trailer brake torques (for details please refer to section 24.11.1 'Brake - General')
Tr.Buffer<pos>.Frc.q<i>			N	Buffer force due to the generalized coordinate <i>
Tr.Buffer<pos>.Frc_tot			N	Total buffer force
Tr.Buffer<pos>.l Tr.Buffer<pos>.l _com Tr.Buffer<pos>.l_ext Tr.Buffer<pos>.l_kin		Fr1	m	Buffer length at <pos> total by compliance external, offset by kinematics
Tr.Camber<pos> Tr.CamberTwin<pos>	Trailer.Tire[i].Camber Trailer.TwinTire[i].Camber	Fr1	rad	Camber angle (ISO 8855:2011, 7.1.17)
Tr.Con.ax_1 Tr.Con.ay_1 Tr.Con.az_1	Trailer.ConBdy1.a_1[0] Trailer.ConBdy1.a_1[1] Trailer.ConBdy1.a_1[2]	Fr1	m/s^2	Translational acceleration of center of mass in vehicle frame
Tr.Con.ax Tr.Con.ay Tr.Con.az	Trailer.ConBdy1.a_0[0] Trailer.ConBdy1.a_0[1] Trailer.ConBdy1.a_0[2]	Fr0	m/s^2	Translational acceleration of center of mass in global frame
Tr.Con.tx Tr.Con.ty Tr.Con.tz	Trailer.ConBdy1.t_0[0] Trailer.ConBdy1.t_0[1] Trailer.ConBdy1.t_0[2]	Fr0	m	Translation of center of mass in global frame
Tr.Con.v	Trailer.ConBdy1.v		m/s	Velocity of center of mass
Tr.Con.vHori	Trailer.ConBdy1.vHori		m/s	Horizontal frailer velocity (ISO 8855:2011, 5.1.5) $Tr.\text{vHori} = \sqrt{Tr.\text{vx}^2 + Tr.\text{vy}^2}$
Tr.Con.vx_1 Tr.Con.vy_1 Tr.Con.vz_1	Trailer.ConBdy1.v_1[0l Trailer.ConBdy1.v_1[1] Trailer.ConBdy1.v_1[2]	Fr1	m/s	Translational velocity of center of mass in vehicle frame
Tr.Con.vx Tr.Con.vy Tr.Con.vz	Trailer.ConBdy1.v_0[0] Trailer.ConBdy1.v_0[1] Trailer.ConBdy1.v_0[2]	Fr0	m/s	Translational velocity of center of mass in global frame
Tr.C<pos>.AxleFrc		Fr1	N	Total vertical force on axle <pos>
Tr.C<pos>.C.t_0.x Tr.C<pos>.C.t_0.y Tr.C<pos>.C.t_0.z		Fr0	m	Translation of carrier center point at <pos> in Fr0

Name UAQ	Name C-Code	Frame	Unit	Info (data type, if not double)
Tr.C<pos>.C.v_0.x Tr.C<pos>.C.v_0.y Tr.C<pos>.C.v_0.z		Fr0	m/s	Velocity of carrier center point at <pos> in Fr0
Tr.C<pos>.Frc2C_1.x Tr.C<pos>.Frc2C_1.y Tr.C<pos>.Frc2C_1.z		Fr1	N	Total force to the wheel carrier at <pos> in vehicle frame
Tr.C<pos>.Frc2C_2.x Tr.C<pos>.Frc2C_2.y Tr.C<pos>.Frc2C_2.z		Fr2	N	Total force to the wheel carrier at <pos> in wheel frame
Tr.C<pos>.Frc2CExt_0.x Tr.C<pos>.Frc2CExt_0.y Tr.C<pos>.Frc2CExt_0.z		Fr0	N	External forces to the wheel carrier, defined in global frame
Tr.C<pos>.Frc2CExt_1.x Tr.C<pos>.Frc2CExt_1.y Tr.C<pos>.Frc2CExt_1.z		Fr1	N	External forces to the wheel carrier, defined in vehicle frame
Tr.C<pos>.Frc2CExt_2.x Tr.C<pos>.Frc2CExt_2.y Tr.C<pos>.Frc2CExt_2.z		Fr2	N	External forces to the wheel carrier, defined in wheel frame
Tr.C<pos>.GenFrc0 Tr.C<pos>.GenFrc1			N	Generalized force at the generalized coordinate <i>
Tr.C<pos>.GenInert0 Tr.C<pos>.GenInert1			kg	Generalized inertia at the generalized coordinate <i>
Tr.C<pos>.l<part>_dq<i>		Fr1	m	Change of the length of <part> at <pos> in direction of the generalized coordinate <i>
Tr.C<pos>.Maggi0.x Tr.C<pos>.Maggi0.y Tr.C<pos>.Maggi0.z		Fr1	-	Elements of the "Maggi-Matrix": Describes the relation between the generalized coordinate q0 and the rotation of the wheel carrier
Tr.C<pos>.Maggi1.x Tr.C<pos>.Maggi1.y Tr.C<pos>.Maggi1.z		Fr1	-	Elements of the "Maggi-Matrix”: Describes the relation between the generalized coordinate q1 and the rotation of the wheel carrier
Tr.C<pos>.P.t_0.x Tr.C<pos>.P.t_0.y Tr.C<pos>.P.t_0.z Tr.C<pos>.Twin.P.t_0.x Tr.C<pos>.Twin.P.t_0.y Tr.C<pos>.Twin.P.t_0.z		Fr0	m	Coordinates of wheel road contact point P
Tr.C<pos>.q<i> Tr.C<pos>.q<i>p Tr.C<pos>.q<i>pp			m m/s m/s^2	Translation of wheel carrier at <pos> in direction of the generalized coordinate <i>
Tr.C<pos>.rx_com Tr.C<pos>.ry_com Tr.C<pos>.rz_com		Fr1	rad	Rotation of carrier <pos> at reference point by compliance with rotation sequence ZYX
Tr.C<pos>.rx_dq<i> Tr.C<pos>.ry_dq<i> Tr.C<pos>.rz_dq<i>			rad/m	Rotation of wheel carrier at <pos> in direction of the generalized coordinate <i
Tr.C<pos>.rx_ext Tr.C<pos>.ry_ext Tr.C<pos>.rz_ext		Fr1	rad	Rotation of carrier <pos> at reference point external, offset with rotation sequence ZYX
Tr.C<pos>.rx_kin Tr.C<pos>.ry_kin Tr.C<pos>.rz_kin		Fr1	rad	Rotation of carrier <pos> at reference point by kinematics with rotation sequence ZYX
Tr.C<pos>.rx Tr.C<pos>.ry Tr.C<pos>.rz	Trailer.SuspMod[i].r_zxy[0] Trailer.SuspMod[i].r_zxy[1] Trailer.SuspMod[i].r_zxy[2]	Fr1	rad	Total rotation of carrier <pos> at reference point with rotation sequence ZXY
Tr.C<pos>.rxv_ext Tr.C<pos>.ryv_ext Tr.C<pos>.rzv_ext		Fr1	rad/s	External rotational velocity of the carrier reference point with rotation sequence ZXY
Tr.C<pos>.rxv Tr.C<pos>.ryv Tr.C<pos>.rzv		Fr1	rad/s	Rotational velocity of the carrier reference point by kinematics with rotation sequence ZXY
Tr.C<pos>.Trq2C_1.x Tr.C<pos>.Trq2C_1.y Tr.C<pos>.Trq2C_1.z		Fr1	Nm	Total torque to the wheel carrier at <pos> in vehicle frame
Tr.C<pos>.Trq2C_2.x Tr.C<pos>.Trq2C_2.y Tr.C<pos>.Trq2C_2.z		Fr2	Nm	Total torque to the wheel carrier at <pos> in wheel frame
Tr.C<pos>.Trq2CExt_1.x Tr.C<pos>.Trq2CExt_1.y Tr.C<pos>.Trq2CExt_1.z		Fr1	Nm	External torques to the wheel carrier at <pos> defined in Fr1

Name UAQ	Name C-Code	Frame	Unit	Info (data type, if not double)
Tr.C<pos>.Trq2CExt_2.x Tr.C<pos>.Trq2CExt_2.y Tr.C<pos>.Trq2CExt_2.z		Fr2	Nm	External torques to the wheel carrier at <pos> defined in Fr2
Tr.C<pos>.TrqGyro_2.x Tr.C<pos>.TrqGyro_2.z		Fr2	Nm	Gyroscopic Trq to carrier at <pos> defined in Fr2
Tr.C<pos>.Trq_T2W Tr.C<pos>.Twin.Trq_T2W			Nm	Tire torque around wheel spin axis at <pos>
Tr.C<pos>.tx_com Tr.C<pos>.ty_com Tr.C<pos>.tz_com		Fr1	m	Translation of carrier reference point at <pos> by compliance
Tr.C<pos>.tx_dq<i> Tr.C<pos>.ty_dq<i> Tr.C<pos>.tz_dq<i>		Fr1	m	Translation of wheel carrier at <pos> in direction of the generalized coordinate <i> Example: Tr.CFL.tz_dq0
Tr.C<pos>.tx_ext Tr.C<pos>.ty_ext Tr.C<pos>.tz_ext		Fr1	m	Translation of carrier reference point at <pos> external, offset
Tr.C<pos>.tx_kin Tr.C<pos>.ty_kin Tr.C<pos>.tz_kin		Fr1	m	Translation of carrier reference point at <pos> by kinematics
Tr.C<pos>.tx Tr.C<pos>.ty Tr.C<pos>.tz	Trailer.SuspMod[i].t[0] Trailer.SuspMod[i].t[1] Trailer.SuspMod[i].t[2]	Fr1	m	Total translation of carrier reference point at <pos>
Tr.C<pos>.txv_ext Tr.C<pos>.tyv_ext Tr.C<pos>.tzv_ext		Fr1	m/s	External velocity of the carrier reference point
Tr.C<pos>.txv Tr.C<pos>.txv Tr.C<pos>.txv		Fr1	m/s	Velocity of the carrier reference point by kinematics
Tr.Damp<pos>.Frc_tot.q<i>			N	Damper force due to the generalized coordinate <i>
Tr.Damp<pos>.Frc Tr.Damp<pos>.Frc_ext Tr.Damp<pos>.Frc_tot			N	Damper force internal external total
Tr.Damp<pos>.l Tr.Damp<pos>.l_com Tr.Damp<pos>.l_ext Tr.Damp<pos>.l_kin			m	Damper length total by compliance external, offset by kinematics
Tr.Damp<pos>.v			m/s	Damper velocity
Tr.Fr1.ax Tr.Fr1.ay Tr.Fr1.az	Trailer.Fr1_a_0[0] Trailer.Fr1_a_0[1] Trailer.Fr1_a_0[2]	Fr0	m/s^2	Translational acceleration of Fr1 in global frame
Tr.Fr1.rvx Tr.Fr1.rvy Tr.Fr1.rvz	Trailer.Fr1_rv_zyx[0] Trailer.Fr1_rv_zyx[1] Trailer.Fr1_rv_zyx[2]	Fr0	rad	Trailer rotation speed whereby Tr.rz = Tr.Yaw.ry = Car.Pitch and Car.rz = Car.Roll (ISO 8855:2011, 5.2.1, 5.2.2 and 5.2.3)
Tr.Fr1.rx Tr.Fr1.ry Tr.Fr1.rz	Trailer.Fr1_r_zyx[0] Trailer.Fr1_r_zyx[1] Trailer.Fr1_r_zyx[2]	Fr0	rad	Trailer rotation angles, Cardan angles, whereby Tr.rz = Tr.Yaw.ry = Car.Pitch and Car.rz = Car.Roll (ISO 8855:2011, 5.2.1, 5.2.2 and 5.2.3)
Tr.Fr1.tx Tr.Fr1.ty Tr.Fr1.tz	Trailer.Fr1_t_0[0] Trailer.Fr1_t_0[1] Trailer.Fr1_t_0[2]	Fr0	m	Position of Fr1 in global frame
Tr.Fr1.vx Tr.Fr1.vy Tr.Fr1.vz	Trailer.Fr1_v_0[0] Trailer.Fr1_v_0[1] Trailer.Fr1_v_0[2]	Fr0	m/s	Velocity of Fr1 in global frame
Tr.Fx<pos> Tr.Fy<pos> Tr.Fz<pos> Tr.FxTwin<pos> Tr.FyTwin<pos> Tr.FzTwin<pos>	Trailer.Tire[i].Frc_W[0] Trailer.Tire[i].Frc_W[1] Trailer.Tire[i].Frc_W[2] Trailer.TwinTire[i].Frc_W[0] Trailer.TwinTire[i].Frc_W[1] Trailer.TwinTire[i].Frc_W[2]	FrW	N	Longitudinal, lateral and vertical ground reaction force at wheel/road contact point
Tr.Hitch.dr_0.x Tr.Hitch.dr_0.y Tr.Hitch.dr_0.z		Fr0	rad	Rotational angle difference between vehicle and trailer in the hitch point
Tr.Hitch.dt_0.x Tr.Hitch.dt_0.y Tr.Hitch.dt_0.z		Fr0	m	Position difference between vehicle and trailer in the hitch point
Tr.Hitch.Frc2Tr.x_1 Tr.Hitch.Frc2Tr.y_1 Tr.Hitch.Frc2Tr.z_1		Fr1/ FrDrw	N	Hitch force from car to trailer in hitch frame (Fr1 for semi trailer or FrDrw for drawbar trailer)

Name UAQ	Name C-Code	Frame	Unit	Info (data type, if not double)
Tr.Hitch.Frc2TrExt.x_1 Tr.Hitch.Frc2TrExt.y_1 Tr.Hitch.Frc2TrExt.z_1	Trailer.Hitch.Frc2TrExt_1[0] Trailer.Hitch.Frc2TrExt_1[1] Trailer.Hitch.Frc2TrExt_1[2]	Fr1/ FrDrw	N	External, user defined hitch force to trailer in hitch frame (Fr1 for semi trailer or FrDrw for drawbar trailer)
Tr.Hitch.Frc2Tr.x Tr.Hitch.Frc2Tr.y Tr.Hitch.Frc2Tr.z	Trailer.Hitch.Frc2Tr_0[0] Trailer.Hitch.Frc2Tr_0[1] Trailer.Hitch.Frc2Tr_0[2]	Fr0	N	Total hitch force from car to trailer in global frame
Tr.Hitch.Trq2Tr.x_1 Tr.Hitch.Trq2Tr.y_1 Tr.Hitch.Trq2Tr.z_1		Fr1/ FrDrw	Nm	Hitch torque from car to trailer in hitch frame (Fr1 for semi trailer or FrDrw for drawbar trailer)
Tr.Hitch.Trq2TrExt.x_1 Tr.Hitch.Trq2TrExt.y_1 Tr.Hitch.Trq2TrExt.z_1	Trailer.Hitch.Trq2TrExt_1[0] Trailer.Hitch.Trq2TrExt_1[1] Trailer.Hitch.Trq2TrExt_1[2]	Fr1/ FrDrw	Nm	External, user defined hitch torque to trailer in hitch frame (Fr1 for semie trailer or FrDrw for drawbar trailer)
Tr.Hitch.Trq2Tr.x Tr.Hitch.Trq2Tr.y Tr.Hitch.Trq2Tr.z	Trailer.Hitch.Trq2Tr_0[0] Trailer.Hitch.Trq2Tr_0[1] Trailer.Hitch.Trq2Tr_0[2]	Fr0	Nm	Total hitch torque from car to trailer in global frame
Tr.Hitch.tx Tr.Hitch.ty Tr.Hitch.tz	Trailer.Hitch.t_0[0] Trailer.Hitch.t_0[1] Trailer.Hitch.t_0[2]	Fr0	m	Hitch position in global frame
Tr.Hitch.vx Tr.Hitch.vy Tr.Hitch.vz	Trailer.Hitch.v_0[0] Trailer.Hitch.v_0[1] Trailer.Hitch.v_0[2]	Fr0	m/s	Hitch velocity in global frame
Tr.InclinAngle<pos> Tr.InclinAngleTwin<pos>	Trailer.Tire[i].InclinAngle TrailerTwinTire[i].InclinAngle.	FrW	rad	Inclination angle of wheel at <pos> (ISO 8855:2011, 7.1.16)
Tr.LongSlip<pos> Tr.LongSlipTwin<pos>	Trailer.Tire[i].LongSlip Trailer.TwinTire[i].LongSlip		–	Longitudinal slip of wheel <pos>
Tr.Pitch	Trailer.Pitch		rad	Trailer pitch angle (ISO 8855:2011, 5.2.2) Positive, if front goes down and rear of trailer comes up
Tr.PitchAcc	Trailer.PitchAcc		rad/s^2	Pitch angle acceleration (ISO 8855: 2011, 5.2.2)
Tr.PitchVel	Trailer.PitchVel		rad/s	Pitch angle velocity (ISO 8855:2011, 5.2.2)
Tr.Roll	Trailer.Roll		rad	Trailer roll angle (ISO 8855:2011, 5.2.3) Positive, if right side goes down and left side comes up
Tr.RollAcc	Trailer.RollAcc		rad/s^2	Roll angle acceleration (ISO 8855:2011, 5.2.3)
Tr.RollVel	Trailer.RollVel		rad/s	Roll angle velocity (ISO 8855:2011, 5.2.3)
Tr.SecSpring<pos>.Frc_tot			N	Secondary spring force
Tr.SecSpring<pos>.Frc_tot.q0			N	Generalized force for the coordinate q0 due to the secondary spring force
Tr.SideSlipAngle	Trailer.ConBdy1.SideSlipAngle		rad	Sideslip angle (ISO 8855:2011, 5.2.9)
Tr.SideSlipAngle2	Trailer.ConBdy1.SideSlipAngle2		rad	Sideslip angle with an offset of 2*PI
Tr.SideSlipAngleVel	Trailer.ConBdy1.SideSlipAngleV- el		rad/s	Sideslip angle velocity (ISO 8855:2011, 5.2.9)
Tr.SimPhase			-	Simulation phase (integer)
Tr.SlipAngle<pos> Tr.SlipAngleTwin<pos>	Trailer.Tire[i].SlipAngle Trailer.TwinTire[i].SlipAngle		rad	Slip angle at <pos>
Tr.Spring<pos>.Frc_tot.q<i>			N	Spring force due to the generalized coordinate <i>
Tr.Spring<pos>.Frc Tr.Spring<pos>.Frc_ext Tr.Spring<pos>.Frc_tot			N	Spring force at <pos> internal external total
Tr.Spring<pos>.l Tr.Spring<pos>.l _com Tr.Spring<pos>.l _ext Tr.Spring<pos>.l _kin			m	Spring length at <pos> total by compliance external, offset by kinematics
Tr.Stabi<pos>.Frc_tot.q<i>			N	Stabilizer force due to the generalized coordinate <i>
Tr.Stabi<pos>.Frc Tr.Stabi<pos>.Frc_ext Tr.Stabi<pos>.Frc_tot			N	Stabilizer force at <pos> internal external total
Tr.Stabi<pos>.l Tr.Stabi<pos>.l_com Tr.Stabi<pos>.l_ext Tr.Stabi<pos>.l_kin			m	Stabilizer length at <pos> total by compliance external, offset by kinematics
Tr.TrqAlign<pos> Tr.TrqAlignTwin<pos>	Trailer.Tire[i].Trq_W[2] Trailer.TwinTire[i].Trq_W[2]	FrW	Nm	Aligning torque in the tire road contact point

Name UAQ	Name C-Code	Frame	Unit	Info (data type, if not double)
Tr.Trq_Ext2Whl<pos>	Trailer.Tire[i].Trq_Ext2W		Nm	External torque around wheel spin axis. Internally it is used as follows: Wheel_rota = (Trq_Brake2Wheel + Trq_Ext2Wheel + c->Trq_Tire2W) / Iyy
Tr.TrqOvert<pos> Tr.TrqOvertTwin<pos>	Trailer.Tire[i].Trq_W[0] Trailer.TwinTire[i].Trq_W[0]	FrW	Nm	Overturning torque in the tire road contact point
Tr.TrqRoll<pos> Tr.TrqRollTwin<pos>	Trailer.Tire[i].Trq_W[1] Trailer.TwinTire[i].Trq_W[1]	FrW	Nm	Rolling torque in the tire road contact point
Tr.TurnSlip<pos> Tr.TurnSlipTwin<pos>	Trailer.Tire[i].TurnSlip Trailer.TwinTire[i].TurnSlip		1/m	Turn slip at <pos>
Tr.tx Tr.ty Tr.tz	Trailer.ConBdy1.t_0[0] Trailer.ConBdy1.t_0[1] Trailer.ConBdy1.t_0[2]	Fr0	m	Trailer translation in global frame
Tr.v	Trailer.ConBdy1.v		m/s	Trailer velocity
Tr.Virtual.Frc_0.x Tr.Virtual.Frc_0.y Tr.Virtual.Ftrc_0.z	Trailer.Virtual.Frc_0[0] Trailer.Virtual.Frc_0[1] Trailer.Virtual.Frc_0[2]	Fr0	N	External virtual force acting to trailer body in global frame
Tr.Virtual.Frc_1.x Tr.Virtual.Frc_1.y Tr.Virtual.Frc_1.z	Trailer.Virtual.Frc_1[0] Trailer.Virtual.Frc_1[1] Trailer.Virtual.Frc_1[2]	Fr1	N	External virtual force acting to trailer body in vehicle frame
Tr.Virtual.Trq_0.x Tr.Virtual.Trq_0.y Tr.Virtual.Trq_0.z	Trailer.Virtual.Trq_0[0] Trailer.Virtual.Trq_0[1] Trailer.Virtual.Trq_0[2]	Fr0	Nm	External virtual torque acting to trailer body in global frame
Tr.Virtual.Trq_1.x Tr.Virtual.Trq_1.y Tr.Virtual.Trq_1.z	Trailer.Virtual.Trq_1[0] Trailer.Virtual.Trq_1[1] Trailer.Virtual.Trq_1[2]	Fr1	Nm	External virtual torque acting to trailer body in vehicle frame
Tr.v<pos> Tr.vTwin<pos>	Trailer.Tire[i].v Trailer.TwinTire[i].v		m/s	Wheel velocity (based on wheel rotation and wheel radius)
Tr.vx Tr.vy Tr.vz	Trailer.ConBdy1.v_0[0] Trailer.ConBdy1.v_0[1] Trailer.ConBdy1.v_0[2]	Fr1	m/s	Trailer velocity in vehicle frame
Tr.WheelSpd_<pos> Tr.WheelSpd_Twin<pos>	Trailer.Tire[i].WheelSpd Trailer.TwinTire[i].WheelSpd		rad/s	Wheel rotation speed at <pos>
Tr.WheelTurnSpd_<pos>	Trailer.Tire[iS].WheelTurnSpd Trailer.TwinTire[i].WheelTurnSpd		rad/s	Wheel turning speed at <pos>
Tr.W<pos>.Radius Tr.W<pos>.Twin.Radius	Trailer.Tire[i].WRadius Trailer.TwinTire[i].WRadius		m	Wheel radius at <pos>
Tr.W<pos>.rot	TrailerTire[i].rot.		rad	Wheel rotation around wheel spin axis at <pos>
Tr.Yaw	Trailer.Yaw		rad	Yaw angle (ISO 8855:2011, 5.2.1) Angle between X-axis of trailer and X-axis of earth fixed system. Positive for positive rotation around Z-axis
Tr.YawAcc	Trailer.YawAcc		rad/s^2	Yaw angle acceleration (ISO 8855:2011, 5.2.1)
Tr.YawVel	Trailer.YawVel		rad/s	Yaw angle velocity (ISO 8855:2011, 5.2.1)

24.15.2 Trailer Load

Name UAQ	Name C-Code	Name CM4SL	Frame	Unit	Info (data type, if not double)
Tr.Load.<i>.mass	Trailer.Load[i].mass	Trailer Load_<i> mass		kg	Mass of load number i
Tr.Load.<i>.tx Tr.Load.<i>.ty Tr.Load.<i>.tz	Trailer.Load[i].tx Trailer.Load[i].ty Trailer.Load[i].tz	Trailer Load_<i> tx Trailer Load_<i> ty Trailer Load_<i> tz	Fr1	m	Position of i-the load in vehicle frame

24.16 Traffic

24.16.1 General

Name UAQ	Name C-Code	Unit	Info (data type, if not integer)
<Nb> :=0, .., n (number)			
Traffic.Freeze	Traffic.Freeze	-	Freeze the motion of all objects (boolean) 0 = no freeze; 1 = freeze
Traffic.nObjs	Traffic.nObjs	-	Total number of traffic objects (integer)
Traffic.TimeOffset	Traffic.TimeOffset	s	Time offset for the global traffic time

24.16.2 Traffic object

The quantities in the column “Simulink“ refer to those with respect to the “TrafficObject“ block in the CarMaker Simulink library.

Name UAQ	Name C-Code	Name Simulink	Frame	Unit	Info (data type, if not integer)
<preQ> := Traffic.<ObjectName> Pointer tTrafficObj *pObj = Traffic_GetByTrfId(<Internal TrafficId>) or Traffic_GetByObjId(<Global ObjId>) FrObj: Traffic object frame					
-	pObj->Tr2Fr0[a][b]	Fr2Fr0 <a> <b>		-	Transformation matrix from FrTraffic to Fr0 a, b = [0..2]
<preQ>.AutoDrv.AxTgt				m/s^2	Target longitudinal acceleration of the traffic autonomous driver
<preQ>.AutoDrv.DesrSpd				m/s	Desired travel speed of the traffic autonomous driver
<preQ>.a_0.x <preQ>.a_0.y <preQ>.a_0.z	pObj->a_0[0] pObj->a_0[1] pObj->a_0[2]	a_0 x a_0 y a_0 z	Fr0	m/s^2	Global acceleration of traffic object reference point (locating in the middle of the rearmost surface)
<preQ>.a_1.x <preQ>.a_1.y <preQ>.a_1.z	pObj->a_1[0] pObj->a_1[1] pObj->a_1[2]	a_1 x a_1 y a_1 z	FrObj	m/s^2	Global acceleration of traffic object reference point (locating in the middle of the rearmost surface)
<preQ>.DetectLevel	pObj->DetectLevel	DetectLevel		-	Detection level by a sensor of object for IPG Movie visualization (integer): 0 = not detected 1 = detected, but not crucial 2 = detected and crucial 3-7 = not set by the sensor, but could be used for user specific applications with different colors
<preQ>.Distance	pObj->Distance			m	Driven distance
<preQ>.GCS.Long <preQ>.GCS.Lat <preQ>.GCS.Elev	pObj->t_GCS.Long pObj->t_GCS.Lat pObj->t_GCS.Elev		GCS	rad rad m	Global position (GCS) of the traffic object's frame origin: longitude, latitude, elevation (by default disabled, can be enabled in SimParameters file; see for details section 3.1.16 'Geographic coordinate system')
<preQ>.JuncObjId <preQ>.nextJuncObjId	pObj->JuncObjId pObj->nextJuncObjId				Actual (last) and next road junction object Id
<preQ>.Lane.Act.* isRight / LaneId	pObj->Lane.Act.* isRight / LaneId				Lane information of the lane, on which the object is currently moving
<preQ>.LatVel	pObj->LatVel	LatVel	FrR	m/s	Lateral velocity relative to route, path y axis (derivation of tRoad)
<preQ>.LinkObjId	pObj->LinkObjId				Actual road link object Id
<preQ>.LongAcc	pObj->LongAcc	LongAcc		m/s^2	Longitudinal acceleration of CoM in course orientation
<preQ>.LongVel	pObj->LongVel	LongVel		m/s	Longitudinal velocity of CoM in course orientation
<preQ>.ManNo					Current minimaneuver Id; can be used as target minimaneuver Id. Special treatment: - if a maneuver start condition is specified and not yet true, ManNo equals -1; setting the ManNo to an value unequal -1 starts the maneuver - if the target minimaneuver Id is greater than the number of minimaneuver steps, the maneuver is terminated - if the object at the end of the maneuver is not deactivated/hidden, the maneuver can be restarted
<preQ>.onJunction	pObj->onJunction				Flag if the traffic object CoM is on junction?
<preQ>.Pitch_X				rad	The pitch angle of the motion model 4Wheel relative to road area

Name UAQ	Name C-Code	Name Simulink	Frame	Unit	Info (data type, if not integer)
<preQ>.Roll_X				rad	The roll angle of the motion model 4Wheel relative to road area
<preQ>.rx <preQ>.ry <preQ>.rz	pObj->r_zyx[0] pObj->r_zyx[1] pObj->r_zyx[2]	r_zyx x r_zyx y r_zyx z	Fr0	rad	Rotation angles of traffic object
<preQ>.rxv <preQ>.ryv <preQ>.rzv	pObj->rv_zyx[0] pObj->rv_zyx[1] pObj->rv_zyx[2]		Fr0	rad/s	Rotational velocity of traffic object
<preQ>.s2lastJunc <preQ>.s2nextJunc	pObj->s2lastJunc pObj->s2nextJunc			m	Road distance along route from CoM to last / next junction
<preQ>.sRoad	pObj->sRoad	sRoad		m	Traffic object road coordinate (detected in the traffic object middle) on route / path
<preQ>.State	pObj->State	State		-	State of traffic object (integer): 0 = motion deactivated, object hidden 1 = motion active, object visible 2 = fixed visible object (e.g: building) 3 = free motion by user
<preQ>.SteerAng				rad	Steering angle of front tire(s) using the motion model 2Wheel or 4Wheel
<preQ>.tRoad	pObj->tRoad	tRoad	FrR	m	Traffic object's lateral distance of CoM to route / path
<preQ>.t2Ref	pObj->t2Ref		FrR	m	Traffic object's lateral distance of CoM referred to road reference line
<preQ>.tx <preQ>.ty <preQ>.tz	pObj->t_0[0] pObj->t_0[1] pObj->t_0[2]	t_0 x t_0 y t_0 z	Fr0	m	Global position of the traffic object reference point (locating in the middle of the rearmost surface)
<preQ>.v_0.x <preQ>.v_0.y <preQ>.v_0.z	pObj->v_0[0] pObj->v_0[1] pObj->v_0[2]	v_0 x v_0 y v_0 z	Fr0	m/s	Global velocity of traffic object reference point (locating in the middle of the rearmost surface)
<preQ>.v_1.x <preQ>.v_1.y <preQ>.v_1.z	pObj->v_1[0] pObj->v_1[1] pObj->v_1[2]		FrObj	m/s	Global velocity of traffic object reference point (locating in the middle of the rearmost surface)

24.17 Traffic Light

Name UAQ	Name C-Code	Unit	Info (type, if not double)
<pre> = TrfLight.<TrafficLightName>			
<pre>.Mode	TrfLight.Objs[Nb].Mode	-	Traffic light running mode (integer): 0: Automatic (defined by phase times) 1: Manual
<pre>.State	TrfLight.Objs[Nb].State	-	Traffic light state/phase (integer): 0: All lights off 1: Green light on 2: Yellow light on 3: Red light on 4: Red-Yellow light on
<pre>.tRemain	TrfLight.Objs[Nb].tRemain	s	Remaining time until next phase

Chapter 25

Start Conditions

25.1 Overview

In the CarMaker GUI use Simulation / Determine Start Values to take a snapshot of the vehicle state at an arbitrary time during a TestRun. CarMaker will store the current vehicle state into the SimOutput/<hostname>/Snapshot.info file.

These values could be used to setup vehicle start conditions .

25.2 Vehicle Start Configuration

Main target: Get all accelerations to zero by a proper start configuration (vehicle state).

The following points are taken into account: loads, vehicle velocity, road conditions at start position (course direction, gradient, slope, ...), suspension compression and forces, tire deflection, vehicle roll and pitch angle etc.

CarMaker's approach is to modify DOFs so that accelerations are minimized. This is done step by step, for one scope after the other, see Table 25.1: .

Table 25.1: Start Condition calculation

Scopes, Vehicle.SimPhases	**Info**	**Quantities, Fcn, etc.**
PrepModel	vehicle at origin of the world (O_1 is near O_0) vehicle stands still no TestRun loads are applied to the vehicle yet only trim loads are active	compressions glsModel () tz (car.Fr1.t_0[2]) rx, ry (car.Fr1.r_zyx_0[0, 1]) 4x q0 (car.Susp?.?.Kin.q0)
PrepModel	loading the vehicle, step by step adding TestRun loads	compressions glsModel ()

Table 25.1: Start Condition calculation

Scopes, Vehicle.SimPhases	Info	Quantities, Fcn, etc.
PrepWhlPlane	vehicle with start speed on the flat road, driving in along +x axis get wheels to zero slip conditions	wheel/rim rotation glsTireSlip () v (car.Fr1.v_0[0]) 4x rotv (Rim.rotv)
PrepOrientation	vehicle on the 3D road at start position	compressions glsModel3D () tz (car.Fr1.t_0[2]) rx, ry (car.Fr1.r_zyx_0[0, 1]) 4x q0 (car.Susp?.?.Kin.q0)
PrepWhlRoad	wheel/rim rotation at start position	wheel/rim rotation glsTireSlip () 4x rotv (car.TFL.IF.Rim_rotv)
PowerTrain Static Cond	....	rotations

25.3 Configure Vehicle Start Conditions

Parameter **VhclStartCond.Kind = <k>**

How does the vehicle model gets into it's state at simulation start?
There are three ways to do so:

- <k> := " " or "default"
  The optimal start conditions are calculated by CarMaker from the given parameterization.
- <k> := "substmodel"
  Linear substitution models are used to calculate start conditions. This is important if your suspension or your tire forces are calculated by a Simulink model, which is only executed at simualtion state "running".
  After TestRun starts, CarMaker switches from the substitution model back to the user model. For further information see section 'Kind "substmodel"'.
- <k> := "value"
  The defined constants are used, no optimization is done before TestRun starts. For further information see section 'Kind "value"'.
- <k> := "KinCfg"
  Before the optimal start conditions are calculated by CarMaker, the suspension is pretensioned to make compression zero to an equilibrium state.
  This mode is implemented only for vehicle model Car.

Optional parameter, given in the vehicle file.

Kind "substmodel"

Linear substitution models are used to determine the best start condition.

Parameter **VhclStartCond.Susp<pos>.Frc = <l0> <dF/dq>**
<pos> := FL, FR, RL, RR

Force is calculated by Frc = dF/dq * (l0 - l);
dF/dq has to be unequal to zero, optional parameter.

Kind “value”

The following vehicle start values that are read from the vehicle file are used directly. They are all optional values with 0.0 as default.

Parameter **VhclStartCond.Fr1.v_0 = <v_[km/h]>**
VhclStartCond.Fr1.t_0 = <tx_[m]> <ty_[m]> <tz_[m]>
VhclStartCond.Fr1.r_zyx = <rx_[deg]> <ry_[deg]> <rz_[deg]>
VhclStartCond.Susp<pos>.q = <q0_[m]>

VhclStartCond.Susp<pos>.Spring.l0 = <l0_[m]>
VhclStartCond.W<pos>.rotv = <rotv_[rad/sec]>
<pos> := FL, FR, RL, RR
StartCond.PT.Clutch = <mode> (open, closed, auto (v>v0 -> closed, stand still -> open))

They are ssigned in the function Vehicle_New (). They may be used in the preparation phase.

25.4 Software Interface

Vhcl_StartCond_New ()

- called at end of Vhcl_New () (or at the beginnig of Vhcl_StaticCond_Calc ())
- reads default values
- parametrizes start condition mode

Vhcl_StartCond_Export ()

- called at end of Vhcl_StaticCond_Calc ()
- may be called at end of PowerTrain_StaticCond_Calc ()
- exports results from start condition optimisation to
  - to SimOutput/Project/<...>/StartCond.info
  - Time stamp, test run name, vehicle name, ...
  - to matlab workspace
  - to ...
- Results are
  - DOF’s
  - spring length
  - tire deflection, tire forces

Vhcl_Cfg_Update ()

- Updates knowledge of starting conditions

- Start Velocity -> vCar -> Tire/Rim rotv -> Gear -> Clutch ->...

Appendix A

Data Storage Result File Formats

A.1 IPG erg-files

A.1.1 Type 2 erg-files

Type 2 erg-files were introduced in CarMaker 3.0. As of the writing of this document, this format is the current standard result file format of CarMaker.

The type 2 erg-file format tries to overcome the restriction of its predecessor, that quantity data cannot be stored according to its original data type, but only as single precision floating-point values. An accompanying infofile (see below, section A.1.3) is required in order to determine the record structure as well as name, unit and other properties of the quantities.

File naming conventions

The default filename extension of a type 2 erg-file is “.erg”.

The accompanying infofile has the same name as the erg-file, with the extension “.erg.info”. Alternatively, only “.info” as extension is also allowed.

File structure

- A type 2 erg-file contains a 16 byte binary file header, followed by a sequence of binary records, each of identical structure and size.
- The file header contains the following data in exactly that order:
  - An 8 byte format identifier;
    its value is the string “CM-ERG”, padded with zeros.
  - A 1 byte unsigned integer value specifying the version number of the file format;
    currently this value is 1.
  - A 1 byte boolean value specifying whether the byte order of the file is big-endian;
    0 means little-endian, everything else means big-endian.
  - A 2 byte unsigned integer specifying the record size (in the respective byte order).
  - 4 unused bytes, reserved for use in future versions;
    all these bytes are zero.

- Each record contains the quantity values $v_1 ... v_n$ for quantities $q_1 ... q_n$ respectively (a "time step") and stored according to each quantity's specifiy data type as denoted in the accompanying infofile.
- Inside each record the quantity values are grouped according to their data type and ordered by decreasing size of their data type, i.e. all doubles first, then floats etc., char/uchar last. The record is padded with zero bytes to make the record size an integer multiple of 8.
- The sequence of all values v_i comprises the complete data vector for quantity q_i.

Implementation considerations

- The format identifier string together with the version number provide enough information to unambiguously identify a type 2 erg-file.
- The record size must match the record size computed from the accompanying infofile, otherwise the erg-file is to be rejected.
- An incomplete record at the end of the file is not considered an error and should simply be discarded.

Example

The following example describes the first few records of a version 2 erg-file containing 3 quantities of different data types.

Record size is 16 bytes = 1 x sizeof(double) + 1 x sizeof(float) + 1 x sizeof(char) + 3 fill bytes.

```
Bytes 0 - 7:    char[8]         Id ("CM-ERG\0\0")      Header
Byte  8:        unsigned char   Version (1)
Byte  9:        unsigned char   Byte-order (0, i.e. little endian)
Bytes 10 - 11:  unsigned short  Record size (13)
Bytes 12 - 15:  char[4]         Reserved (0)

Bytes 16 - 23:  double          Value #1 of q1         Record #1
Bytes 24 - 27:  float           Value #1 of q2
Byte  28:       char            Value #1 of q3
Bytes 29 - 31:  zeroes          Padding

Bytes 32 - 39:  double          Value #2 of q1         Record #2
Bytes 40 - 43:  float           Value #2 of q2
Byte  44:       char            Value #2 of q3
Bytes 45 - 47:  zeroes          Padding

Bytes 48 - 55:  double          Value #3 of q1         Record #3
Bytes 56 - 59:  float           Value #3 of q2
Byte  60:       char            Value #3 of q3
Bytes 61 - 63:  zeroes          Padding

Bytes 64 - 71:  double          Value #4 of q1         Record #4
...
```

The accompanying infofile might e.g. have the following contents:

```
File.Format = erg
File.ByteOrder = LittleEndian

File.At.1.Name = Time
File.At.1.Type = Double
Quantity.Time.Unit = s

File.At.2.Name = Car.v
File.At.2.Type = Float
Quantity.Car.v.Unit = m/s
```

```
File.At.3.Name = DM.Shifting
File.At.3.Type = Char
Quantity.DM.Shifting.nStates = 2

File.At.4.Name = $none$
File.At.4.Type = 3 Bytes
```

A.1.2 Type 1 erg-files

Type 1 erg-files (also called “fortran unformatted” erg-files) were standard in CarMaker releases earlier than 3.0. The quantity values contained in such a file can be read without additional information, but name, unit and other properties of the quantities can only be found in an accompanying infofile (see section A.1.3).

Please note: This format is outdated and should no longer be used.

File naming conventions

The default filename extension of a type 1 erg-file is “.erg”.

The accompanying infofile has the same name as the erg-file, with the extension “.erg.info”. Alternatively, only “.info” as extension is also allowed.

File structure

- A type 1 erg-file is a sequence of binary records, each of identical structure and size.
- Each record contains the following elements, in exactly this order:
  - An initial 4 byte integer record delimiter, containing the record size.
  - n single precision, 4 byte floating-point values $v_1 ... v_n$ for quantities $q_1 ... q_n$ respectively (a “time step”).
  - A terminating 4 byte integer record delimiter, repeating the record size.
- The record size stored in the record delimiters is the number of floating point values in the vector, multiplied by 4 (the size in bytes of each value). The number of floating point values in a record must be at least 1 and at most 16383.
- The sequence of all values v_i comprises the complete data vector for quantity q_i.

Implementation considerations

- The limitation about the number of floating point values in a record allows for easy detection of the byte order used in the file, as the most significant 16 bits of the record size are always zero and the least significant 16 bits are always non-zero.
- In order to identify a file as a type 1 erg-file, simply check whether the record size read from the first 4 bytes of the file can also be found as the record delimiters of the next few records.
- The record size must be divisible by 4. If this is not the case, the file must be rejected. The number of quantities in each record can then be computed easily.
- An incomplete record at the end of the file is not considered an error and should simply be discarded.

Example

The following example describes the first few records of a version 1 erg-file containing 3 quantities. Record size is always 12 (3 quantities of 4 bytes size each).

```
Bytes 0 - 3:     int          Record size (12)        Record #1
Bytes 4 - 7:     float        Value #1 of q1
Bytes 8 - 11:    float        Value #1 of q2
Bytes 12 - 15:   float        Value #1 of q3
Bytes 16 - 19:   int          Record size (12)

Bytes 20 - 23:   int          Record size (12)        Record #2
Bytes 24 - 27:   float        Value #2 of q1
Bytes 28 - 31:   float        Value #2 of q2
Bytes 32 - 35:   float        Value #2 of q3
```

```
Bytes 36 - 39:  int             Record size (12)

Bytes 40 - 43:  int             Record size (12)          Record #3
Bytes 44 - 47:  float           Value #3 of q1
Bytes 48 - 51:  float           Value #3 of q2
Bytes 52 - 55:  float           Value #3 of q3
Bytes 56 - 59:  int             Record size (12)

Bytes 60 - 63:  int             Record size (12)          Record #4
...
```

The accompanying infofile might e.g. have the following contents:

```
File.Format = FORTRAN_Binary_Data
File.ByteOrder = LittleEndian

File.At.1.Name = Time
File.At.1.Type = Float
Quantity.Time.Unit = s

File.At.2.Name = Car.vx
File.At.2.Type = Float
Quantity.Car.vx.Unit = m/s

File.At.3.Name = Car.ax
File.At.3.Type = Float
Quantity.Car.ax.Unit = m/s*s
```

A.1.3 Content of the accompanying Infofile

Both type 1 and type 2 erg-files keep additional information in an accompanying Infofile. In addition to the format-relevant entries listed below, such an infofile is also allowed to contain other, application-specific entries.

General infofile entries

File.Format = *format*

Specifies the file format of the erg-file.

Legal values for *format* are:

erg – The file is a type 2 erg-file.
FORTRAN_Binary_Data – The file is a type 1 erg-file.

Example

```
File.Format = erg
```

File.ByteOrder = *byteorder*

Specifies the byte order of the data stored in the erg-file.

Legal values for *byteorder* are:

LittleEndian – The data in the file is in little-endian format, i.e. low byte first.
BigEndian – The data in the file is in big-endian format, i.e. high byte first.

Example

```
File.ByteOrder = LittleEndian
```

Quantity-specific infofile entries

File.At.<*i*>.Name = *name*
File.At.<*i*>.Type = *type*

Positional information about the quantity data in each record of the erg-file. Specifies that value v_i in each data record is stored in the file as the given type and belongs to the quantity with the given name.

The *name* is an arbitrary string. It should not contain whitespace, brackets ([ ]), colons (:), equal signs (=), backslashes (\ ) and other characters with a special meaning in infofiles. Furthermore the characters in *name* should be restricted to 7-bit ASCII.

The following table lists what is allowed for *type*. Most of the basic C types are supported.

Value	Description
Double	8 byte double precision floating-poing value

Value	Description
Float	4 byte single precision floating-point value
LongLong	8 byte signed integer value
ULongLong	8 byte unsigned integer value
Long	4 byte signed integer value (deprecated, synonymous with *Int*)
ULong	4 byte unsigned integer value (deprecated, synonymous with *UInt*)
Int	4 byte signed integer value
UInt	4 byte unsigned integer value
Short	2 byte signed integer value
UShort	2 byte unsigned integer value
Char	1 byte signed integer value
UChar	1 byte unsigned integer value
n Bytes	*n* bytes of unknown data to be skipped (only allowed in conjunction with *name* = $none$, see below)

For type 1 erg-files the type *Float* is implicit, so this entry needs not to be specified.
For type 2 erg-files the type specification is mandatory.

Implementation considerations

- In order to determine the number of quantities in the file, try reading the value of successive *File.At.<i>.Name* entries and stop at the first non-existing entry. All remaining *File.At.<i>...* entries possibly present in the file should be ignored.
- For type 1 erg-files adding up the size of the (implicit) type of each quantity should yield the exact record size stored in the record delimiters of the erg-file. Otherwise the file must be rejected.
- For type 2 erg-files adding up the size of the type of each quantity should yield the exact record size stored in the erg-file's file header. Otherwise the file must be rejected.
- If *name* is specified as *$none$*, this means that the data for this quantity is to be skipped in each record. In the case of type 2 erg-files the entry for *type* must be present; additionally the type may also be specified as a number of bytes. In the case of type 1 erg-files exactly 4 bytes are to be skipped. In general any type entry specifying only a number of bytes implies that the data is to be skipped (even if *name* is not *$none$*).

Examples

```
File.At.1.Name = Time
File.At.1.Type = Double

File.At.2.Name = $none$
File.At.2.Type = 6 Bytes
```

Quantity.*<name>*.Unit = *unit*

Optional. A string specifying the physical unit of the quantity. Digital quantities in particular have no unit associated with them. For a list of commonly used unit names see menu entry ***Help / Unit List*** in IPGControl.

Examples

```
Quantity.Time.Unit = s
```

```
Quantity.Car.v.Unit = m/s
```

Quantity.*<name>*.nStates = *n*
Quantity.*<name>*.FirstState = *s*

Optional. If specified, qualifies the quantity as a so-called digital quantity with only *n* distinct values ("states") ranging from *s* to *s+n-1*. If no entry for the first state is given, values are assumed to start at 0.

Visualization software such as IPGControl can make use of this information in order to optimize the display of quantities.

Examples

```
Quantity.GearNo.nStates = 7      # Backwards = -1, 0 = idle, 1-5 = forward
Quantity.GearNo.FirstState = -1
```

Quantity.*<name>*.Offset = *b*
Quantity.*<name>*.Factor = *k*

Optional. When reading the quantity data, each resulting value v of the quantity will automatically be computed as $v = v_{orig} * k + b$, with v_{orig} being the original value read from the file. This only applies to non-digital quantities. If not specified, the factor *k* defaults to 1 and the offset *b* defaults to 0.

Examples

```
Quantity.Measured1.Factor = 42.5
Quantity.Measured1.Offset = 1024
```

Anim.Msg.<No>.Time = *value*
Anim.Msg.0.Class = *Anim*
Anim.Msg.0.Id = *ID*
Anim.Msg.0.Data = *binary converted data*
Anim.VehicleClass = *type*
Anim.Vehicle.MovieSkin = *path/to/.obj*

Information required for IPGMovie to replay the simulation based on the result file.

Testrun = *relative/path/to/TestRun*

Name and relative path to the TestRun used in the simulation.

NamedValues.Count = *number names*
KeyValues.Count = *number names*

List of *Named Values* and *Key Values* in case they were used in the simulation (see User's Guide, section 14.5 'Variable Types').

Comment: ***Text***

Optional: Comment added to the result info file.

CarMaker.Version = ***CarMaker Version Number***
CarMaker.Version.Apo = ***APO Version Number***
CarMaker.Version.Info = ***Infofile Version Number***
CarMaker.Version.Driver = ***IPGDriver Version Number***
CarMaker.Version.Road = ***IPGRoad Version Number***
CarMaker.Version.Tire = ***IPGTire Version Number***

Version numbers of CarMaker and its modules such as the APO communication service, Infofile library, IPGDriver, IPGRoad, IPGTire.

A.1.4 Measurement Data Format (MDF)

The standardized data file format *Measurement Data Format (MDF)* can be used by various post-processing tools. This format accepts data based on a maximum of three different sample rates.

Two different compressed MDF variants, *MDF zipped* and *MDF transposed*, can also be generated by CarMaker.

Appendix B

Traffic Sign File Format

B.1 Traffic sign file format

Traffic signs are defined by their traffic sign infofile. The traffic sign infofiles are recognized by the .trfsign extension and the *FileIdent* entry. The filename is also used as internal name for the traffic sign. For more readability an alias name for the sign can be set (see below). The traffic sign infofile must be stored in a (country) subfolder inside the *TrafficSigns* folder. This folder should be named after a country indicator according to ISO-Code 3166. The *TrafficSigns* folder is a top level folder inside a DataPool. Traffic signs in a DataPool with higher priority shadow traffic signs of the same name in DataPools with lower priority.

B.1.1 Traffic sign infofile entries

FileIdent = TrafficSign *<version>*

Specifies the infofile type and version. Current version is 1.0.

Example

```
FileIdent = TrafficSign 1.0
```

Info.de = *info*
Info.en = *info*

Optional. Additional information about the traffic sign can be stored in the Info key. Right now this key is not read.

Example

```
Info.de = 101 (StVO)
```

Name.de = *name*
Name.en = *name*

Specifies the sign name as it appears in the GUI. The traffic sign name can be set for both languages which are supported by the CarMaker GUI.

Example

```
Name.de = Gefahrenstelle
Name.en = Warning
```

TexFile = *texturefile*

A graphic file which contains the texture printed on the front of the sign. Allowed formats are png and jpg. The texture file must be stored in the same folder as the traffic sign infofile.

Example

```
TexFile = Caution.png
```

TexFile.Back = *texturefile*

Optional. A graphic file which contains the texture printed on the back side of the sign. Allowed formats are png and jpg.

Class = *class*

Specifies the sign class. The class is used for a coherent presentation of the traffic signs in the GUI. Traffic signs of the class supplement can only be chosen as a supplement sign.

Known classes are:

- Advisory
- Warning
- Regulatory
- Miscellaneous
- Supplement
- Other

Examples

```
Class = Warning
```

Shape = *shape*

Specifies the shape of the sign.

Known shapes are:

- Circular

- Rectangular
- TriangularUp
- TriangularDown
- Square
- Square45
- Octagonal

Examples

```
Shape = TriangularUp
```

Sizes = *sizename0 sizename1 ...*

Specifies a list of size-names for a sign. For each size-name a size must be specified. The first listed size-name will be used as default.

Examples

```
Sizes = M S L
```

Size.<sizename> = *width height*

Specifies the width and height for the size *sizename.*

Examples

```
Size.S = 0.42 0.42
Size.M = 0.60 0.60
Size.L = 0.75 0.75
```

SizeBase = *sizename*

Optional. In case SizeBase is set the given texture is not scaled to the actual chosen size. In case the actual chosen size is bigger than the size given by SizeBase the texture will be repeated to cover the sign plate.

Examples

```
SizeBase = S
```

Val0.Name.de = *name0*
Val0.Name.en = *name0*
Val1.Name.de = *name1*
Val1.Name.en = *name1*

Optional. Specifies the name for the value fields. A traffic sign can have up to 2 values which can be parametrized in the GUI.

Examples

```
Val0.Name.de = Geschwindigkeit
Val0.Name.en = Speed
```

Val0.Unit.de = *unit0*
Val0.Unit.en = *unit0*
Val1.Unit.de = *unit1*
Val1.Unit.en = *unit1*

Optional. Specifies the unit for the the given sign values.

Example

```
Val0.Unit.de = km/h
Val0.Unit.en = kph
```

Movie.tclgeo:
tclcmd1
tclcmd2
...

Optional. This options allows to use a set of tcl commmands to print text or lines on the sign.

Besides the standard tcl-commands there are three special tcl-commands that can be used.

Writing text on the sign:

```
DrawText text ?options?
text: the text which is written on the sign
options:
   - font <font>                    default: sans
   - position <yOff> <zOff>         default: 0 0
   - width <w>
   - height <h>
   -justified <j>                   default: center
   -rotation <xRot>                 default: 0
   -back                            default: not set, writes on the backside of the sign
```

Actual known fonts are:

- sans
- sans-bold
- lcd
- din1451 (used for german traffic signs)
- CaractersL1 (used for french traffic signs)
- CaractersL2
- UK_Transport (used for traffic signs of the UK)
- UK_TransportHeavy

Allowed values for justification are:

- left
- center
- right

Getting the width of a text:

```
GetTextWidth font text
```

Drawing lines:

```
DrawLine thickness ?options?
   thickness                    line thichness
   -length <length>             default: width of bounding box of the traffic sign
   -back                        default: not set, draws the line on the backside of the sign
   -rotation <rx>               default: 0
   -translation <tx> <ty> <tz>  default: tx=0 ty=0 tz=0
```

Caution: rotation and translation is done in the order they are listed

There are also some variables that can be accessed:

```
# $Val0      refers to Value 0 given in the GUI
# $Val1      refers to Value 1 given in the GUI
# $Width     refers to sign with for the chosen size
# $Height    refers to sign height for the chosen size
# $Size      refers to the chosen size
```

Example

```
Movie.tclgeo:
   set txt "[expr {int($Val0)} ]"
   set txtLength [GetTextWidth din1451 $txt]
   set h [expr {0.67*$Height}]
   set w [expr {0.9*$h/$txtLength}]
   set y [expr {-0.31*$h}]
   DrawText $txt -f din1451  -w $w  -h $h  -j center -position 0 $y
```

Appendix C

Traffic Signs Overview

C.1 France (FRA)

Advisory

Pedestrian crossing

Pedestrian crossing (twosided)

Speed limit

Speed limit end

Warning

Caution

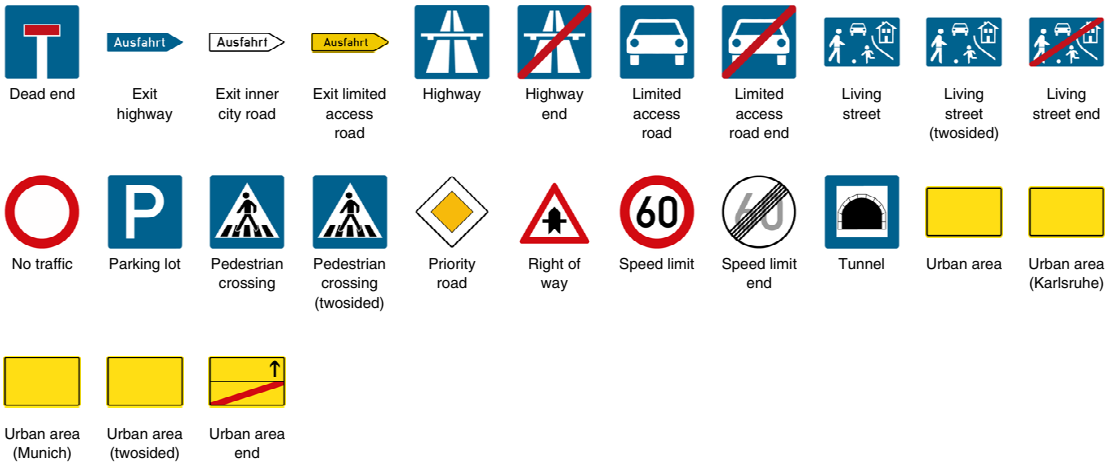
Regulatory

Give way

Stop

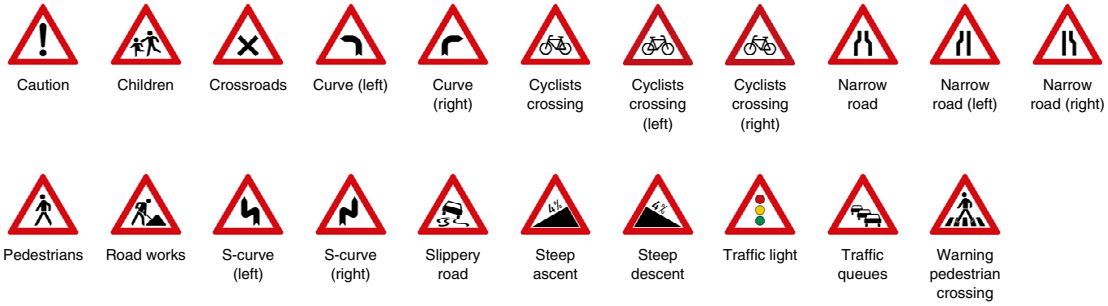
C.2 Germany (DEU)

Advisory



Dead end, Exit highway, Exit inner city road, Exit limited access road, Highway, Highway end, Limited access road, Limited access road end, Living street, Living street (twosided), Living street end, No traffic, Parking lot, Pedestrian crossing, Pedestrian crossing (twosided), Priority road, Right of way, Speed limit, Speed limit end, Tunnel, Urban area, Urban area (Karlsruhe), Urban area (Munich), Urban area (twosided), Urban area end

Warning

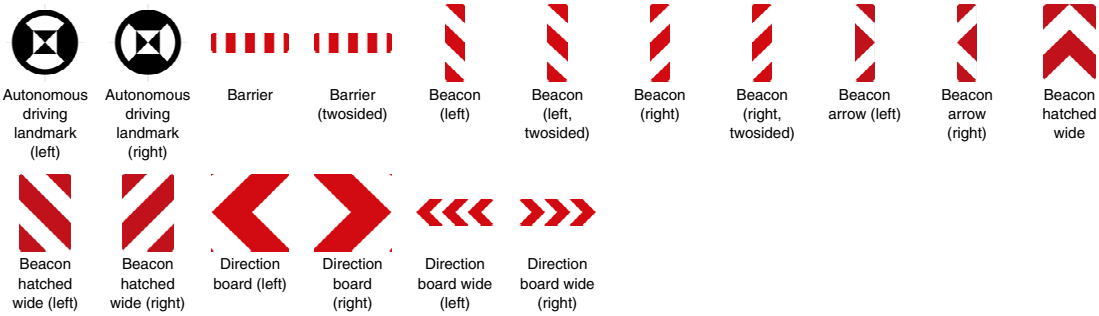


Caution, Children, Crossroads, Curve (left), Curve (right), Cyclists crossing, Cyclists crossing (left), Cyclists crossing (right), Narrow road, Narrow road (left), Narrow road (right), Pedestrians, Road works, S-curve (left), S-curve (right), Slippery road, Steep ascent, Steep descent, Traffic light, Traffic queues, Warning pedestrian crossing

Regulatory

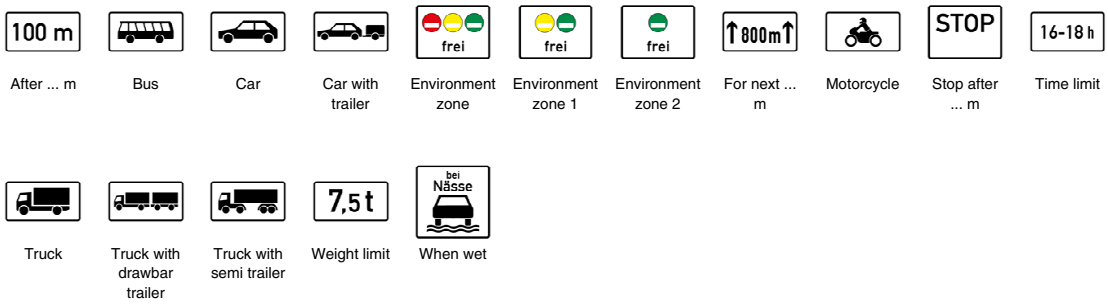
Axle load limit, End of limitations, Environment zone, Environment zone (twosided), Environment zone end, Give way, Height limit, Length limit, Minimum speed, Minimum speed end, No entry, No overtaking, No overtaking end, No overtaking trucks, No overtaking trucks end, Pass (left), Pass (right), Roundabout, Speed limit zone, Speed limit zone (twosided), Speed limit zone end, Stop, Straight or left, Straight or right, Turn (left), Turn (right), Turn ahead (left), Turn ahead (right), Weight limit, Width limit

Miscellaneous



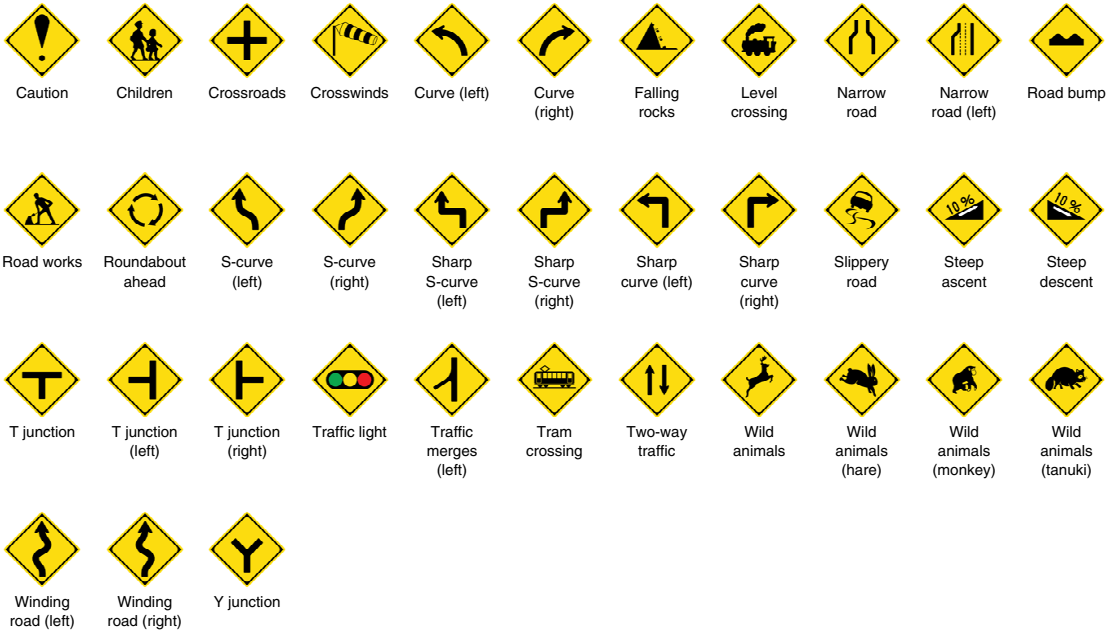
Other

Supplement

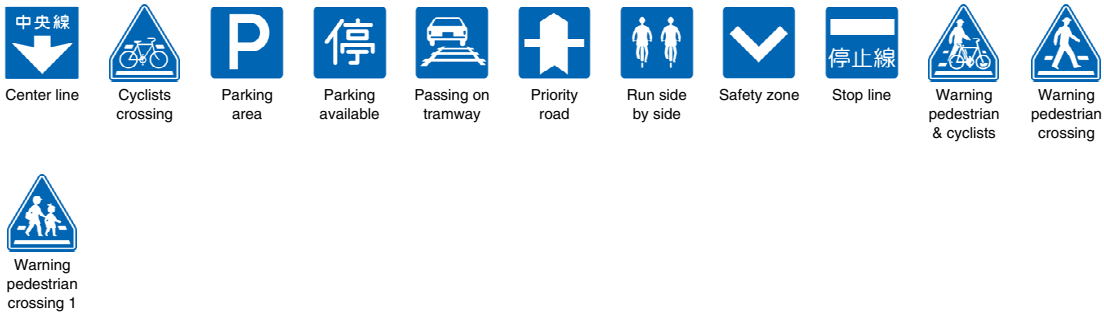


C.3 Japan (JPN)

Warning



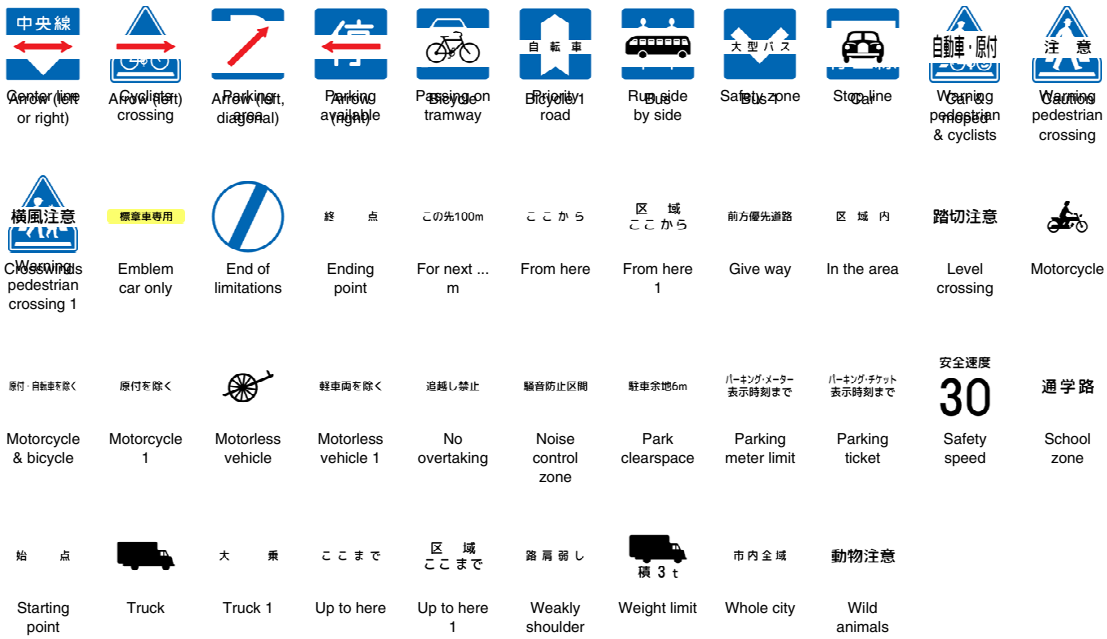
Indication



Regulatory

Bicycle path
Bus lane
Bus lane 1
Direction to be followed
Follow direction (3 lanes)
Follow direction (left)
Follow direction (straight or left)
Follow direction (straight)
Give way
Give way 1
Height limit
Honk horn
Minimum speed
No U-turn
No automotives
No bicycles
No buses
No dangerous substances
No entry
No light vehicles
No motor vehicles
No motorcycles
No motorless vehicles
No overtaking
No parking
No pedestrians
No pedestrians crossing
No stopping
No traffic
No traffic 1
No trucks
No turn (right)
No two right turn mopeds
No two-person motorcycles
One-way street
One-way street (left)
One-way street (right)
One-way street bicycles
One-way street bicycles (left)
Only vehicles
Parking (diagonal)
Parking (longitudinal)
Parking (vertical)
Pass (left)
Pass (right)
Pedestrian path
Roundabout
Shared use path
Speed limit
Stop
Stop 1
Stop 2
Straight
Straight or left
Straight or right
Truck lane
Truck lane 1
Truck lane 2
Turn ahead (left or right)
Turn ahead (left)
Turn ahead (right)
Two-stage right turn for mopeds
Weight limit
Width limit

Indication
Supplement



C.4 United Kingdom of Great Britain and Northern Ireland (GBR)

Advisory

Speed limit Speed limit end

Regulatory

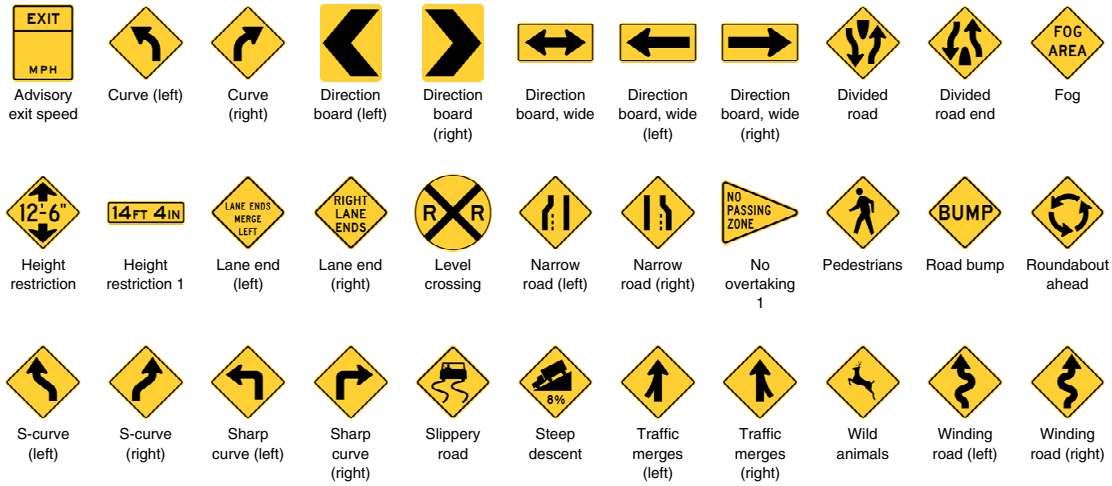
Give way Stop

Warning

Caution

C.5 United States of America (USA)

Warning

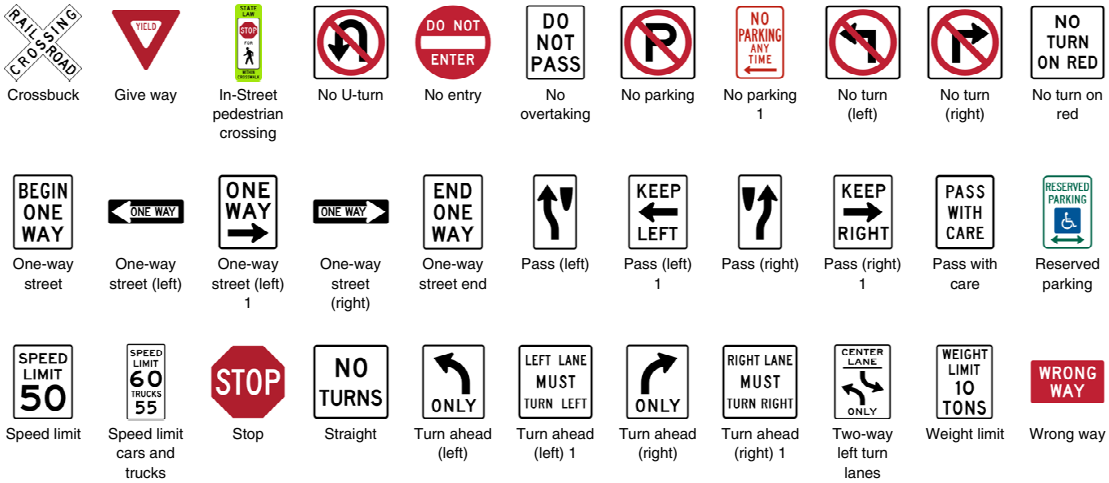


Advisory exit speed | Curve (left) | Curve (right) | Direction board (left) | Direction board (right) | Direction board, wide | Direction board, wide (left) | Direction board, wide (right) | Divided road | Divided road end | Fog

Height restriction | Height restriction 1 | Lane end (left) | Lane end (right) | Level crossing | Narrow road (left) | Narrow road (right) | No overtaking 1 | Pedestrians | Road bump | Roundabout ahead

S-curve (left) | S-curve (right) | Sharp curve (left) | Sharp curve (right) | Slippery road | Steep descent | Traffic merges (left) | Traffic merges (right) | Wild animals | Winding road (left) | Winding road (right)

Regulatory



Crossbuck | Give way | In-Street pedestrian crossing | No U-turn | No entry | No overtaking | No parking | No parking 1 | No turn (left) | No turn (right) | No turn on red

One-way street | One-way street (left) | One-way street (left) 1 | One-way street (right) | One-way street end | Pass (left) | Pass (left) 1 | Pass (right) | Pass (right) 1 | Pass with care | Reserved parking

Speed limit | Speed limit cars and trucks | Stop | Straight | Turn ahead (left) | Turn ahead (left) 1 | Turn ahead (right) | Turn ahead (right) 1 | Two-way left turn lanes | Weight limit | Wrong way

Guide

Distance (3 lines)

Entrance HOV

Entrance HOV 1

Exit

Exit HOV

Exit parking area

Food

Gas station

Handicapped accessible

Hospital

Interstate business loop

Interstate route

Interstate route (3 digits)

Lodging

Milepost

Milepost (3 digits)

Rest area

Rest area 1

State route

Street name

US route

US route (3 digits)

Weather info

School

School bus stop

School zone

Speed limit school zone

Temporary Control Warning

Be prepared to stop

Flagger

Road works

Index

A

Ambient 49
Anti-Roll Bar 128
Archive 48
axis systems 18

B

Brake
- Att.dp2dv 485
- Boo.63Prcnt 472
- Boo.ampli 470, 471, 472, 474, 475
- Boo.delay 472
- Boo.pGrad.mapping 475
- Boo.pMax 475
- Boo.relF 473
- Boo.runOut . . . 470, 472, 474, 475
- Boo.sign2press 472
- Boo.type 469
- Booster 474
- Brake.PedalForce2pMC 462
- Brake.PedalPos2pMC 462
- Brake.pMC_based_on 462
- Brake.pWB2Trq 462
- CircuitConfig 466
- Cyl_.pv.mapping 482
- InCheckV_f.qOri 499
- InCheckV_f.qPipe 499
- Inlet_f.deltaT.mapping 495
- Inlet_f.qOri 497, 498
- Inlet_f.qPipe 499
- Inlet_f.transfer.mapping 494, 495
- LiqTemperature 506
- Low Pressure Accumulator . . 504
- LPA.pMax 484
- LPA.pMin 484
- LPA.vMax 484
- LPAcheckV.qPipe 504
- Master Cylincer 476
- MC.area 476
- MC.closeComp 477
- MC.springConst 477
- MC.springLoad 477
- nue1 506
- nue2 507
- Outlet_f.qOri 500
- Outlet_f.qPipe 500
- Pedal.ratio 467
- Pist_.area 479
- Pist_.ratio 479, 480
- Pist_.rBrake 479
- PLim.pOpen 502
- PLim.qOri 501
- PLim.qPipe 501
- Pump.cLoss 487
- Pump.Full 488
- Pump.genVmax 489
- Pump.p63Prcnt 488
- Pump.pEdge 488
- Pump.qMax 487
- Pump.Zero 488
- PV.qOri 501
- PV.qPipe 501
- PVcheckV.qOri 502
- RefTemperature 506
- SuppL.dp2dv 485
- SV.qOri 503
- SV.qPipe 503
- T1 506
- T2 507

Brake.HandTorque.Value 463
Brake.Kind 52, 450, 453
Brake.TorqueAmplify 450
Brake Booster 474
Brake System
“TrqDistrib” 462
Buffers 123
Bumpers 123

C

Camber 136
Car Model
Force Elements 112
Kinematics and Compliance . 134
Mass Geometry 84
Suspension Anti-Roll Bar 128
Suspension Roll Stabilizer . . . 128
Compliance 134

D

Damper 118
Description 27
design configuration 80
DIN 70000 17

E

equilibrium 80
External Forces 113
ExtSuspFrcs.FName 113
ExtSuspFrcs.Kind 113

F

FileCreator 27
Force Element 112
Fr0 18
Fr1 18
Fr2 19
FrD 19
FrX 19

G

Generalized Coordinates 139

I

ISO 8855 17

K

Kinematics 134

M

Mass Geometry 84
Master Cylinder 476
MESA VERDE 79
Model
Trailer 108, 580, 702

N

Naming Conventions 22

O

OutputQuantities 47

P

primary coordinates 135
PVcheckV.qPipe 502

Q

Quantities 23

R

Roll Stabilizer 128

S

secondary coordinates 135
SimParameters 47
SI-quantities 17
Spin 137
Spring Force 115
start-off configuration 81
SuspF.Buf_Pull 127
SuspF.Buf_Pull.Amplify 126
SuspF.Buf_Pull.tz0 126
SuspF.Buf_Push 125
SuspF.Buf_Push.Amplify 125
SuspF.Buf_Push.tz0 125
SuspF.Damp_Pull 120
SuspF.Damp_Pull.Amplify 120
SuspF.Damp_Push 120
SuspF.Damp_Push.Amplify 119, 122
SuspF.Spring 116, 117, 129, 176
SuspF.Spring.Amplify 116, 117

SuspF.Spring.l0 116
SuspF.Stabi.Amplify 129
SuspR.Buf_Push.tz0 125
SuspR.Damp_Pull 120
SuspR.Damp_Pull.Amplify 120
SuspR.Damp_Push 120
SuspR.Damp_Push.Amplify 119, 122
SuspR.Spring . . . 116, 117, 129, 176, 177, ... 195, 204
SuspR.Spring.Amplify 116, 117
SuspR.Spring.l0 116
SuspR.Stabi.Amplify 129

T

Tire
  Parameters 508
Toe 136
Trailer
  Aerodynamics 596
  Brake System. 593
  Chassis Forces 584
  Hitch 588
  Hitch “Ball” 590, 591
  Hitch “BallDamp”. 591
  Hitch “BallFric” 591
  Hitch “Trapez”. 590
Trailer Model 108, 580, 702
Transformation Matrix 140
Translation 137, 138

U

User Accessible Quantities 107

W

Wheel brakes 478
Wheel compression 138