IPG AUTOMOTIVE

Product Examples Version 7.1.5

CarMaker®

SOLUTIONS FOR VIRTUAL TEST DRIVING

Contents

1 About the Product Examples 5

2 Basic Functions 6

2.1 Driver 6
2.2 DVA 12
2.3 Failsafe Tests 16
2.4 Maneuver 17
2.4.1 Minimaneuver command 17
2.4.2 Open and Closed Loop 19
2.4.3 Record and Replay 21
2.4.4 Special Maneuvers 24
2.5 Movie 25
2.6 Road 28
2.6.1 QuickStartGuide 28
2.6.2 Highway 29
2.6.3 Networks 30
2.6.4 SampledRoads 32
2.6.5 Surface 33
2.7 RTExpressions 37
2.8 Sensors 39
2.9 TestAutomation 46
2.9.1 ScriptControl 46
2.9.2 TestManager 47
2.10 Traffic 53
2.11 VehicleModel 56

2.11.1 Trailer 56

3 Driver Assistance 58

3.1 QuickStartGuide 58

3.2 EuroNCAP 59

4 Powertrain 64

4.1 DrivingCycles 64

4.1.1 Overview 69

4.1.2 Details on Implementation 79

4.1.3 Details on Gear Selection process for WLTC 89

4.2 DrivingScenarios 99

4.3 ElectricAndHybrids 106

4.4 PerformanceTests 107

4.5 PowertrainControl 111

5 Vehicle Dynamics 115

5.1 QuickStartGuide 115

5.2 Longitudinal Dynamics - Braking 115

5.3 Lateral Dynamics - Handling 119

5.4 Stability Control 129

5.5 Rollover 133

5.6 Vertical Dynamics - Ride 138

Chapter 1

About the Product Examples

This document is intended to guide you through several use cases and to demonstrate the ways CarMaker is going to approach them.

The examples provided with the installation illustrate the usage of the software from different perspectives. You will find four sections: a general one for basic functions and three specific ones for the application fields Driver Assistance, Powertrain and Vehicle Dynamics.

- Basic Functions

The folder BasicFunctions provides TestRuns focusing on single CarMaker functions and explains their usage. The *reference* provides you with more background information. Working through all examples should give you a good overview about the CarMaker functionality.

- Driver Assistance

The section DriverAssistance covers different use cases in the field of driver assistance systems, automated driving and active safety. The focus is on building specific scenarios for different applications, e.g. lane support systems or Euro NCAP test cases and the usage of sensor models such as different camera options, e.g. fisheye lenses with distortions.

- Powertrain

The section Powertrain contains applications with different hybrid architectures, use cases such as fuel consumption, drive cycles as well as Real Driving Emissions. Testing the interplay of component parts in the architecture of hybrid cars as well as the operational strategies in real driving situations completes the package.

- Vehicle Dynamics

The section VehicleDynamics includes tests for handling dynamics in different open- and closed- loop maneuvers as well as tests for stability control systems and rollover. This section has a special emphasis on the interaction between the driver model and the vehicle model.

Chapter 2
Basic Functions

2.1 Driver

BackAndForth

The longitudinal controller of IPGDriver includes the possibility of driving forwards and backwards as well as of stopping the vehicle. The TestRun demonstrates different driving and stopping maneuvers of the IPGDriver. The maneuver can be set up via *Main GUI > Parameters > Maneuver > Longitudinal Dynamics*.

ChangingLanes

IPGDriver will follow the road with the selected "Track Offset" relative to the reference line. In the maneuver dialog, you can define the offset in the lateral dynamics control panel (see Figure 2.1). The IPGDriver will perform the offset during the defined time or distance. The acceleration limits given by the g-g diagram (see *GUI > Parameters > IPGDriver*) have no influence on the maneuver; the maneuver has a higher priority.

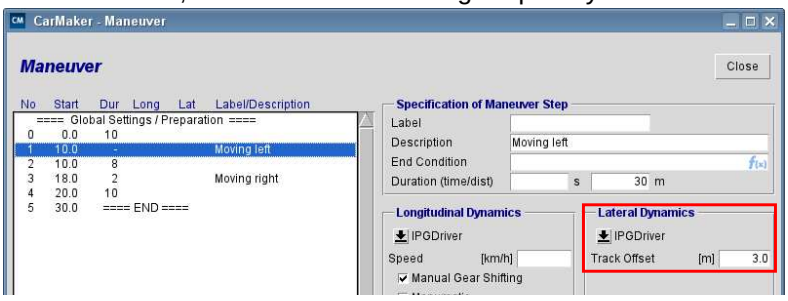


Figure 2.1: Lane Change maneuver by tack offset

Drifting

The IPGDriver offers further options for optimizing and adapting the driver abilities via the interface *Main GUI > Parameters > Driver > Misc. /Additional Parameters*. The additional parameters are listed in the IPGDriver Manual section 3.4 'Misc. / Additional Parameters'.

The TestRun presents the function

- rally driving mode and the corresponding parameters Long.Rally_Active, Long.Rally_BrakeSlipCoef and Long.Rally_SideSlipCoef.

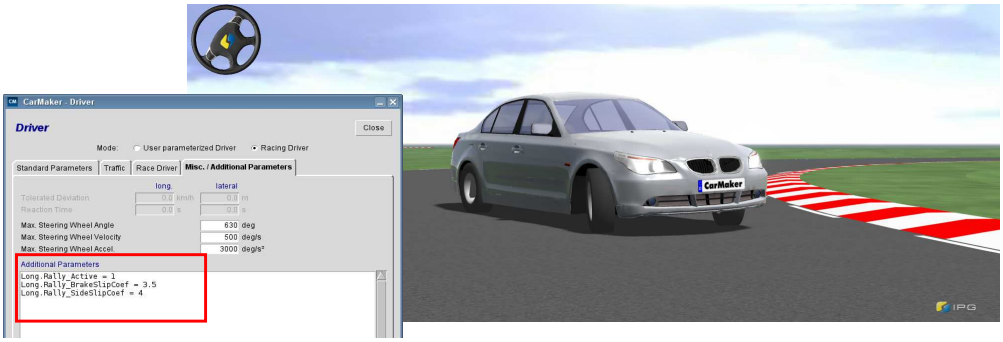


Figure 2.2: Sliding through curves

The rally driving function of IPGDriver allows to "sliding through curves" as shown in Figure 2.2. This driving condition is achieved by increasing the accelerator and brake pedal position at the same time. This leads to braking on the front axle and oversteering on the rear axle due to a superposition of the braking and driving torques. For this reason, this function can only be applied to vehicles with rear-wheel drive.

Please find a more detailed explanation of the application and parameterization of the rally driving mode in section 5.16 'Use Case: Rally Driving - Sliding through curves' in the IPGDriver Manual.

GearShifting.ts

The TestSeries illustrates the manipulation of the driver's shifting behavior. The TestRun "HandlingCourse" is used for the demonstration of the different gear shifting parameters.

The TestSeries shows the following feature:

- different "Gear shifting strategies" shown by a Test Manager example

The TestRun "HandlingCourse" is prepared with different NamedValues for changing the shifting strategy via Test Manager variations named "Low engine speeds", "High engine speeds", "Relaxed" and "Sportive". The diagram in Figure 2.3 shows the different shifting strategies. For example, the low engine speeds driver shifts up if the engine speed exceeds 2000 rpm (see 1. in Figure 2.3).

Figure 2.3: Gear shifting behavior

GeneralSettings.ts

The IPGDriver setting enables the individual modification of the driver characteristics. The parameter presented in the following TestSeries demonstrates the following features:

- the driver's maximum accelerations
- the time for shifting and the speed range for the engine

Open *Main GUI > Parameters > Driver* to see the Driver interface. Find the description of the editable values in the IPGDriver User Manuel chapter *Standard Parameters*. The "Accelerations" are the limits which the driver will not exceed. A detailed description of the acceleration limits can be found in the IPGDriver User Manual chapter *Use Case: Accelerating to Target Velocity*.

The following parameter has an influence on the driver's behavior when cornering.

The test series demonstrates the following functions:

- conducting a parameter variation by means of the Test Manager
- creating "Named Values" in order to access parameters from the Test Manager
- the function of the parameter Corner Cutting Coefficient (CCC)

The test series includes the selection of the TestRun and vehicle, the generation of the diagram in the test report and the variation of the CCC. The Corner Cutting Coefficient will vary from 0 to 1. The result can be seen in the lateral deviation of the vehicle from the road center line (see Figure 2.4). The smaller the CCC, the closer the driver stays to the center line of the driving lane. If the CCC equals 1, the driver drives to the very edge of the lane. For further information, see the IPGDriver Manual section 2.2 'Choice of Course'.

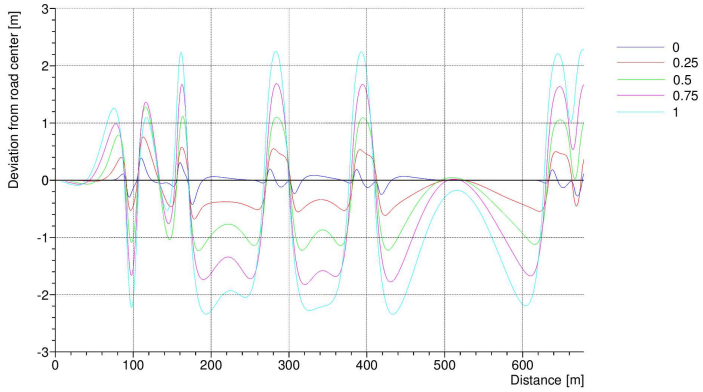


Figure 2.4: Deviation from road center caused by Corner Cutting Coefficient

Before the IPGDriver drives a route, the speed profile is calculated according to the driver parameters. For further information, see the IPGDriver Manual section 2.3 'Choice of Speed'. This TestSeries presents the following features:

- the influence of the parameters Min. dt Accel./Decel.
- the parameter variation in Test Manager
- the creation of Named Values (*Main GUI > Parameter > Driver > Min. dt Accel./ Decel.*)

The test series demonstrates the reaction of the IPGDriver to a modification of the parameter "***Min. dt Accel./Decel.***'." (see Driver interface). The parameter influences the cutting of speed peaks in the static desired speed if the interval between acceleration and deceleration is too short. A smaller parameter usually yields better lap times. The diagram shows the longitudinal acceleration of the vehicle over the driven distance. If the time gap between acceleration and deceleration is smaller than the defined time, IPGDriver will maintain the current speed (see X in diagram Figure 2.5.)

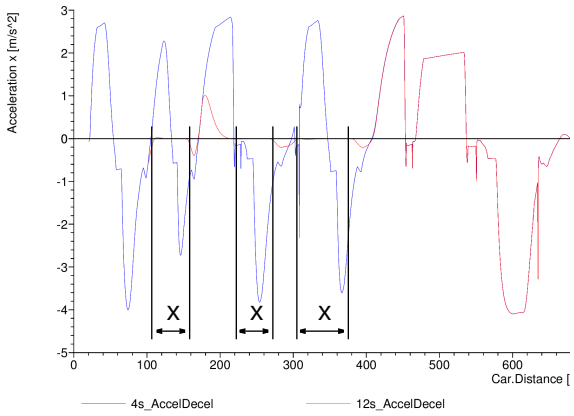


Figure 2.5: IPGDriver Parameter Time gap between acceleration and deceleration

GGDiagram.ts

This TestSeries includes the TestRun HandlingCourse with a variation of the maximum lateral and longitudinal acceleration as well as the combination of these accelerations.

The TestSeries demonstrates the following functions:

- variation of TestRun parameters (NamedValues) by the TestManager
- creation of diagrams in the TestManager

In the TestRun, the varied parameters are set with so-called NamedValues. The first variation illustrates the difference in terms of the maximum lateral acceleration. For the variation Slowly, this is set to 1 m/s2; for the variation Fast, it is set to 5 m/s2. The longitudinal acceleration is illustrated in the diagram in Figure 2.6. The DemoCar does not reach the maximum acceleration of 5 m/s2.

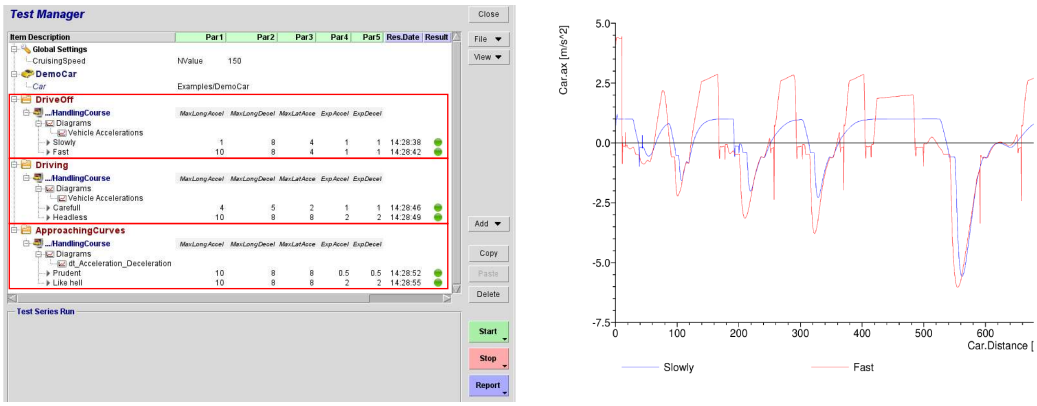


Figure 2.6: Drivers acceleration limits

The second example illustrates the maximum acceleration values for the lateral and longitudinal dynamics. The third example shows a variation of the exponents which describe the maximum permissible acceleration depending on the lateral and longitudinal acceleration, respectively. Find more information in the IPGDriver Manual section 3.1.2 'Accelerations, g-g Diagram'.

HandbrakeUTurn

This TestRun provides a special turning maneuver: a handbrake turn. The entire maneuver consists of manually defined maneuver steps. The features used within this TestRun are:

- the manual control of the longitudinal dynamics
- backward driving
- steer step function for defined steering maneuvers

The successive maneuver steps are tuned exactly for the DemoCar_StrByTrq. When using this TestRun with other vehicles, there may be differences in the trajectory of the vehicle due to differences in the transversal and longitudinal dynamic behavior.

HandlingCourse

The TestRun demonstrates the driver parameters.

The TestRun shows the following:

- a small test track made out of road segments
- NamedValues in the driver parameters for creating TestRun variations in the Test Manager

Figure 2.9 shows the driver parameters user interface (*Main GUI > Parameter > Driver*). The yellow input fields show that NamedValues are already defined here. A NamedValue consists of a "$" followed by the "name" and a default "default value", as in the example: $CruisingSpeed=150 (see 1. in Figure 2.9). This NamedValue can be accessed by TestManager or ScriptControl commands. For more information, see the CarMaker User's Guide chapter *Variable Types*.

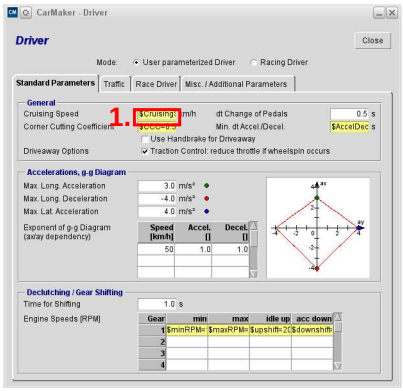


Figure 2.7: Driver parameters

Overtaking

Overtaking is an event-based maneuver. IPGDriver in CarMaker can overtake slower vehicles if enabled by the settings via *Main GUI > Parameters > Driver > Traffic > Overtaking*. The overtaking ratio defines the aggressiveness of the driver. Set to "1", the overtaking will be aggressive and dangerous.

Set to "0", the driver will be careful and overtake only in safe situations. Furthermore, the driver can be parameterized in such a way as to follow a traffic object with a defined distance or time gap.

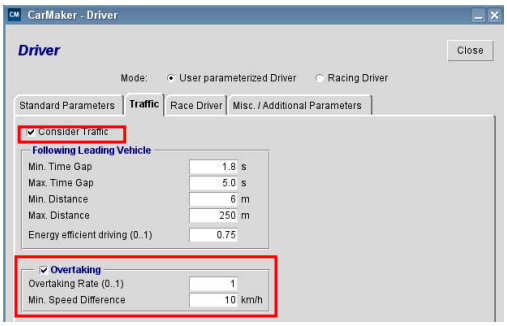


Figure 2.8: Activation of considering traffic and overtaking

RaceDriver

In CarMaker, there are two different types of driver: the "Driver" and the "Race Driver" (see IPGDriver Manual section 3.3 'Race Driver'). The "Driver" is introduced in other TestRuns. This TestRun focuses on the functions of the "Race Driver". Other functions presented here are the following:

- activating the Race Driver
- displaying the lap time

A special characteristic of the Race Driver is lap time optimization. During each lap, the Race Driver tries to optimize the lap time by pushing the maximum transferable force between road and tires to the limits. The cornering force which determines the cornering speed depends on slip and slip angle. The Race Driver tries to maximize these forces by recalculating the parameters for the next curve.

The Race Driver has an additional feature: The lap time is automatically printed in the session log. Figure 2.9 shows a print example of the session log.

```
SIM_START    Examples_new/BasicFunctions/Maneuver/IPGDriver/Driver_RaceMode 2016-07-12
 Time   0.000
SIMULATE     Examples_new/BasicFunctions/Maneuver/IPGDriver/Driver_RaceMode

 Time  96.221
                Lap   1: t=  96.220s Dist=2550.6383m sRoad=1.5172m
 Time 185.742
                Lap   2: t=  89.521s Dist=2558.1965m sRoad=1.4984m
 Time 275.265
                Lap   3: t=  89.523s Dist=2558.2274m sRoad=1.5073m
 Time 364.812
                Lap   4: t=  89.547s Dist=2558.2104m sRoad=1.5071m
 Time 398.753
SIM_END      Examples_new/BasicFunctions/Maneuver/IPGDriver/Driver_RaceMode 398.744s 1
```

Figure 2.9: Session Log in "Race Driver" mode

SnailsPace

The features of this TestRun are:

- building segment-based roads
- setting the "Start Values" in the maneuver interface

The TestRun includes a segment-based road with an s-bend. The driver slows down because of the precalculated "Static Desired Speed" of the course (see the IPGDriver Reference Manual chapter Static Desired Speed).

Via *Main GUI* > *Parameters* > *Maneuver* > *Global Settings / Preparation*, the initial velocity, gear and other parameters can be defined (see Figure 2.10).

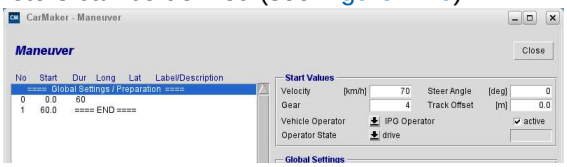


Figure 2.10: Set the initial velocity and gear

SteeringByTorque

to be documented

TrafficStopAndGo

The TestRun demonstrates the driver's following behavior in relation to a lead vehicle.

The TestRun presents the following functions:

- Driver settings in relation to traffic objects
- Creation of speed profiles for traffic objects

Access *Main GUI* > *Parameters* > *Driver* > *Traffic* for the driver settings. For further information, see the IPGDriver Reference Manual chapter *Traffic*. The speed profile of the traffic object is set via *Main GUI* > *Parameters* > *Traffic*. The speed profile of the traffic object is created in the maneuver definition of the object. The absolute time and the corresponding speed is set for this.

2.2 DVA

InteractWithModelInterfaces

CarMaker offers the possibility of applying external forces and torque to the vehicle. This interface is suited for several applications. A disturbing force can be applied to the system, for instance. The external force/torque can also be used for validation tests.

For this, the interfaces Car.Virtual.(Frc/Trq)_0.(x/y/z) with reference to the global system Fr0 and Car.Virtual.(Frc/Trq)_1.(x/y/z) with reference to the vehicle system Fr1 are provided. The forces are always applied to the body's center of gravity.

The interface is expanded with the option of a flexible vehicle body. This allows for the forces/torque to be applied to the individual bodies. For this, the term Virtual is substituted with VirtualB in the name. Car.VirtualB.(Frc/Trq)_0.(x/y/z) and Car.VirtualB.(Frc/Trq)_1.(x/y/z) constitute the interface to the rear part of the vehicle body.

The features of the TestRun are:

- DVA mechanism, which allows you to directly manipulate quantities /states during the simulation
- the minimaneuver command **DVAwr** for automated usage of the DVA mechanism
- the option to apply virtual forces and torques to the body of the vehicle
- the activation of the flexible body via *Vehicle Data Set* > *Vehicle Body* > *Vehicle Body* > *flexible*. This flag enables the flexible body which divides the body into two parts connected by a joint with parameterizable torsional and bending stiffness.

The command below sets the torsional moment around the x-axis of the rear body to -5000 Nm for 10000 ms with a ramp-up time of 8000 ms. See the first slope of the quantity Car.VirtualB.Trq_1.x.

```
DVAwr Car.VirtualB.Trq_1.x AbsRamp 10000 -5000 8000
```

The mode "DVAwr AbsRamp" is used to ramp the force up and down to an absolute value, shown by IPGControl in Figure 2.11. The use of a flexible vehicle body allows for a separate display of the roll angles of the front vehicle body (Car.ConA.Roll_X) and the rear vehicle body (Car.ConB.Roll_X).

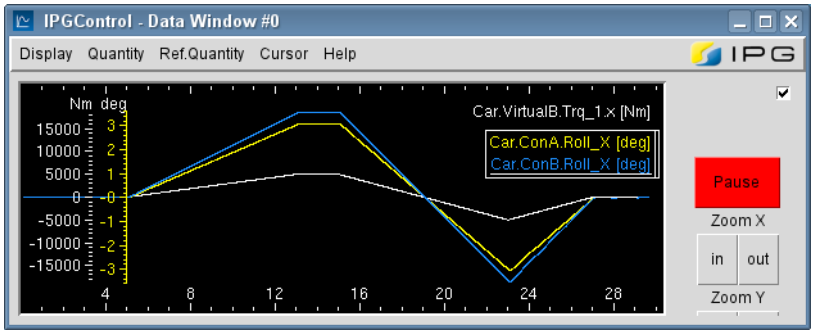


Figure 2.11: IPGControl view of Quantity Car.VirtualB.Trq_1.x and roll angle of body A and B

In the last maneuver all DVA write commands for all quantities are set to zero via the command below. The maneuver duration is set to "0" for a so-called "Last Maneuver" (see TestRun Movie) which will always be executed when the TestRun is finished.

```
DVArel *
```

ManipulateInternalStates

CarMaker offers the possibility to insert disturbances in the system under test, illustrated by a safety test for passenger car and trailer combinations (including caravans and light trailers). The lateral stability test (similar to ISO 8915) for car/trailer combinations determines the damping of the trailer oscillation after the excitement by a steer step.

For this the steering wheel angle is set to a defined value for 1000 ms at a velocity of about 80 km/h. A well tuned car-/trailer combination exhibits a quickly decaying oscillation of the trailer. The reaction can be influenced for example by the geometry of the drawbar and the additional load on the car or trailer, for instance.

The features of the TestRun are:

- DVA mechanism, which allows you to directly manipulate quantities /states during the simulation
- the minimaneuver command **DVAwr** for automatical usage of the DVA mechanism
- car/trailer combination including the hitch of the car and the trailer

The first command shows how to manipulate the steering wheel angle via a **DVA** command. The steer angle is recalculated every cycle. The defined angle of 0.349 rad is maintained for 1000 ms.

```
[DM.ManTime>=15] DVAwr DM.Steer.Ang Abs 1000 0.349
```

Once the maneuver time exceeds 15 seconds, the steering angle is set to 0.349 rad (20 degrees) for 1000 ms. The use of a trailer activates additional quantities, which are abbreviated with Tr. in IPGControl. For the trailer's lateral acceleration Tr.ay, see Figure 2.12.

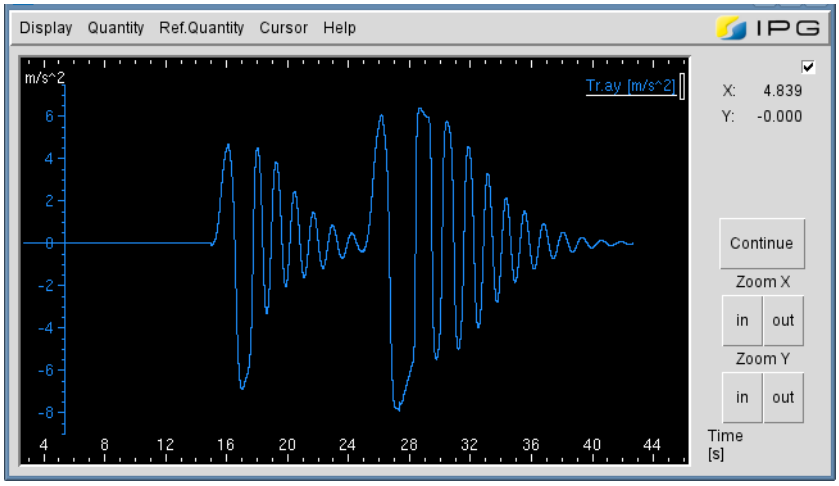


Figure 2.12: Trailer lateral acceleration

In comparison to TestRun VisualizedInteraction the steering wheel value is continuously updated by the DrivMan model, which means, as soon as the DVA intervention ends, the simulation will return to the original values calculated by DrivMan.

For further information about the DVAwr modes, see CarMaker User's Guide.

VisualizedInteraction

This example shows the use of the virtual force Car.Virtual.Frc_1.(x/y/z) as well as the integration of dynamic objects. The vehicle is moved only by the applied force.

The virtual force is applied to a defined point of application (Virtual.PoA) and can be used for different applications. In addition, the force applied to the vehicle is visualized in the simulation's visualization tool "IPGMovie". A dynamic object is used for this, which is modified according to the force. The total force (in x,y,z) influences the dimension of the arrow object. The resulting direction of the force is illustrated by the direction of the arrow. The color of the arrow may be changed as well.

The features of the TestRun are

- DVA mechanism, which allows you to directly manipulate quantities/states during the simulation
- the minimaneuver command DVAwr for automatical usage of the DVA mechanism
- the option to apply virtual forces on the vehicles body and change the force application point
- the insertion of dynamic objects in the IPGMovie visualization.

The force application point is named "Virtual.PoA" and can be set in the additional parameters via *Main GUI* > *Parameters* > *Car* > *Misc.* as follows:

```
Virtual.PoA = <x-coord.> <y-coord.> <z-coord.>
```

The command below applies the virtual force in x-direction (Vehicle Frame, Fr1) to the force application point "Virtual.PoA" of the vehicle.

```
[DM.ManTime>= 5] DVAwr Car.Virtual.Frc_1.x AbsRamp -1 5000 6000
```

If the car has a positive acceleration (Car.ax > 0) the arrows will be colored green; for a negative acceleration (Car.ax < 0) they will be red.

The animated object is modified via a script file (see Installation Directory > /Movie/ VirtualForceAnimation.tclgeo). If you want to modify the animated object, copy it to your local Movie folder in your project directory and modify it according to your requirements using a text editor.

The arrow is visualized using the following OpenGL function:

```
gl Disk <inner_diameter> <outer_diameter> <slices> <stacks>
gl Cylinder <top> <base> <slices> <stacks>
```

The parameter <slices> describes the number of the surface elements around the z-axis (direction of arrow). The parameter <stacks> defines the number of elements along the z-axis. Further steps of the implementation are provided in the file "VirtualForceAnimation.tclgeo".

Figure 2.13: Integration of dynamical objects (for example arrows)

The TestRun shows the different "Modes" of the DVAwr command (Direct Variable Write Access). Each maneuver will demonstrate another "DVAwr Mode" in the following notation

```
DVAwr <QuantName> <Mode> <nCycles> <Val1> [Val2] [nCyclesRamp]
```

The commands can be modified in the maneuver control via *Main GUI* > *Parameters* > *Maneuver*. For further information about the DVAwr modes, see CarMaker User's Guide chapter *DVA: Direct Variable Access*.

Attention! The parameter "Virtual.PoA" is illustrated as the force application point in IPGMovie. If the parameter is modified via "*Main GUI* > *Parameters* > *Car*> *Additional Parameters*", the vehicle has to be saved again. If not, the force application point will not be visualized according to the defined parameters, since the parameters will be read from the info file of the vehicle.

WriteToDedicatedQuantities

One method to create interfaces to models is to define writable quantities, which are linked to some internal model states, which are intended to be manipulated. You can say they are dedicated for manipulation.

The TestRun is a typical use case for rapid prototyping in the concept phase. It demonstrates how easy it is to experiment with powertrain configurations by adding torque to a normally non driven axle in order to identify a better setup.

The functionalites used within this TestRun are the following:

- the DVA mechanism, which allows you to directly manipulate quantities states during the simulation,
- the minimaneuver command DVAwr for automatical usage of the DVA mechanism
- the external torque input interface which enables the addition of external torques (positive/negative) to the rear axle. This can be activated via *Vehicle Data Set* > *Powertrain* > *Driveline* > *External Torque to Differential*.

The DVAwr commands are set in the 'Minimaneuver Commands':

```
[DM.ManTime>=2] DVAwr PT.DL.DriveSrc.1.Trq_ext AbsRamp 500 100 500
[DM.ManTime>=5] DVAwr PT.DL.DriveSrc.1.Trq_ext AbsRamp 500 0 800
```

If the maneuver time is greater than 2 seconds the quantity PT.DL.DriveSrc.1.Trq_ext is ramped up to 100Nm within 500ms. The value is kept until it is changed by the second command (or the Start of a new TestRun), which will return the additional torque to zero within 800ms.

If the duration of the DVAwr command is set to "-1", the value of the variable is maintained "forever" unless it is changed via another DVA command or reset via "DVArel *". Another option is a restart of the application via "Start&Connect".

When setting an unlimited duration of "-1", it is recommended to define a "final maneuver" at the end of the maneuver list with a duration of "0" and the minimaneuver command "DVArel *" in order to avoid any influence of this command on the subsequent simulations.

Anyway, in such a case, often the values are initialized and never written by any internal code. Thus, the manipulation time does not have any effect, the values will remain also after releasing any DVA interaction, because nobody sets the value from the simulation program.

For further information about the DVAwr modes, see CarMaker User's Guide section 'DVAwr - Direct Variable Write Access'.

2.3 Failsafe Tests

Braking_FST_SigChange

This TestRun demonstrates a fail-safe scenario with interchanged wheel speed sensors located front right and front left.

The TestRun includes the following functions:

- CarMaker/HIL signal interchange with FailSafeTester
- Braking on mue-split surface

The signal interchange is set via the minimaneuver commands in maneuver no.0. In the first line, the vehicle operating status is set to *absent* (driver has left the vehicle).

```
1:  VhclOp=absent
2:  FSTChCt WheelSpd.FL.Sig
3:  FSTChCt WheelSpd.FR.Sig
```

In the 2nd and 3rd line, the signals of the front wheel speed sensors are cut using the command *FSTChCt <Sig1>,<Sig2>,... .* For more information see the CarMaker User's Guide chapter *FailSafeTester Mini-Maneuvers Commands.*

With the following commands, the signals are connected. The command *FSTChAd* <Grp> <Ch1>,<Ch2>, ... is followed by the group name and the signals which are connected to the group.

So the `WheelSpd.FL.Sig/i` is connected with `WheelSpd.FR.Sig/o`, forming group *g0.* And the `WheelSpd.FR.Sig/i` is connected with `WheelSpd.FL.Sig/o`, forming group *g1.*

```
4:  FSTChAd g0 WheelSpd.FL.Sig/i
5:  FSTChAd g0 WheelSpd.FR.Sig/o
6:  FSTChAd g1 WheelSpd.FR.Sig/i
7:  FSTChAd g1 WheelSpd.FL.Sig/o
```

The following command has a start condition: if the maneuver time exceeds1.5 seconds, the vehicle operation state request is set to drive.

```
8:  [DM.ManTime>=1.5] VhclOp=drive
```

With the last maneuver, the FailSafeTester is set to the default state via the minimaneuver command *FSTRst.*

This test only has an effect with CarMaker/HIL and an ABS/ESP controller connected with a FailSafeTester.

Braking_FST_SigCut

The TestRun shows a fail-save scenario with signal cut.

The TestRun includes the following functions:

- CarMaker/HIL signal cut with FailSafeTester
- Braking on mue-split surface

In the first maneuver (see *Main GUI > Parameters > Maneuver)* the vehicle accelerates to the driver's *Cruising Speed* of 120 km/h. After 10 s of maneuver time, the signal of **WheelSpd.FR.Sig** is cut (see following command).

```
[DM.ManTime>=10] FSTChCt WheelSpd.FR.Sig
```

A 2 s rolling duration with manual disengaged clutch follows in maneuver no. 2. The braking is also manually controlled in maneuver no. 3. The brake pedal value is set to "1", which corresponds to the full brake pressure.

In maneuver no. 3, the failure is corrected with the following minimaneuver commands:

```
1:  VhclOp=PowerOff
2:  FSTChCn WheelSpd.FR.Sig
3:  [DM.ManTime>4.5] VhclOp=drive
```

The first command sets the vehicle operation state to "power off ". With the function **FSTChCn**, the signal **WheelSpd.FR.Sig** is reconnected. If the maneuver time exceeds 4.5s, the vehicle operation state is requested for the driving state.

With the last maneuver, any signal modification with the FailSafeTester is reset with the command **FSTRst.**

```
[DM.ManTime>=9] FSTRst
```

Further information see User's Guide section 'FSTRst - Reset FST FST2Rst - Reset second FST'.

2.4 Maneuver

2.4.1 Minimaneuver command

JumpToManeuver

The TestRun demonstrates this feature:

- DMjmp command allowing to jump from one maneuver step to another.

The minimaneuver command DMjmp can be used for jumping to maneuvers out of the natural order or to skip maneuvers if some user-defined conditions are true.

```
[Car.v >= (50/3.6)] DMjmp 2
```

Once the car reaches a velocity of 50 kph, the maneuver jump leads to maneuver no. 2.

```
[Car.v >= (80/3.6)] DMjmp Stop
```

You can also name a maneuver using the maneuver "Label" and jump to the maneuver with the defined label, for example "Stop", as shown in the command.

In some cases, it is useful to skip or repeat individual maneuvers. In addition, these maneuver jumps can be triggered by a user-defined event. This allows for the definition of time- or event-controlled maneuvers as well as the implementation of event-controlled functions.

For further information see User's Guide section 'DMjmp - Driving Maneuver Jump'.

PrintLogMessages

This TestRun shows how to write information in the Session Log. The Session Log in CarMaker displays important information during the simulation. The output is further saved in a log file in the folder SimOutput > Log outputs. Moreover, the Session Log can be used for displaying information on unexpected simulation behavior. The Session Log can be used with the Log function in order to print results or information about the running simulation or to stop the simulation when some condition is reached.

The *Session Log* can be written by the following commands:

```
Log "your text"
```

The normal log function displays the text in a normal format.

```
LogWrnS "your text"
```

The warning log opens the Session Log window but the simulation continues:

```
LogErrS "your text"
```

The error Log opens the Session Log window, too, but the simulation stops. For further information see User's Guide section 'Log'.

SaveResults

The DStore functions are special functions to save the simulation results via command. It will save all quantities that are selected in the "Storage of Results" interface (GUI > Application > Output Quantities). By default the results will be saved in the SimOutput folder in the project directory.

To save the results use the command

```
DStoreSave <TimeHistory> <TimeFuture>
```

The command "DStoreSave" is followed by the period of time. The option <TimeHistory> refers to the time passed, while <TimeFuture> displays the future period of time in seconds. To stop the recording of the results, use *DStoreStop*. To remove the result file while *DStoreSave* is in progress, use *DStoreAbort*. The results can be opened with IPGControl.

UseVehicleOperator

Different operating conditions can be described for a vehicle such as "parking" or "driving" and conditions in between which can also be defined depending on the ignition.

The following CarMaker function is used for the simulation of the operating condition:

- Action Command "VhclOp"

With this command, a "request" is sent to CarMaker. If the corresponding general conditions are satisfied, the desired operating condition of the vehicle is triggered. The command

```
VhclOp=<Command>
```

is used to send the operating condition request to the simulation.

The following conditions are defined:

<**absent**> (parked vehicle); <**poweroff**> (ignition off); <**poweracc**> (power for internal loads on); <**poweron**> (engine on); <**drive**> (driving).

For further information about the command options see User's Guide section C.1.4 'Action Commands'.

2.4.2 Open and Closed Loop

ControlledStopping_Accel

IPGDriver can stop the vehicle at a defined distance. Of course, the physical limits of static and dynamic friction cannot be exceeded. In the first maneuver, the TestRun includes an acceleration to 80 kph. The maneuver has the end condition

```
Car.v >= 80/3.6
```

to end the first maneuver when reaching 80 kph. In the second maneuver, the IPGDriver stops the vehicle with the defined deceleration of 8m/s^2.

ControlledStopping_Distance

IPGDriver can stop the vehicle at a defined distance. Of course, the physical limits of static and dynamic friction cannot be exceeded. In the first maneuver, the TestRun includes an acceleration to 80 kph. The maneuver has the end condition

```
Car.v >= 80/3.6
```

to end the first maneuver when reaching 80 kph. In the second maneuver, the IPGDriver stops the vehicle with the defined braking distance of 50 m.

DriveOffNonControlled

The TestRun demonstrates the longitudinal dynamics maneuver of the manual pedal control. It shows the vehicle driving off on a straight road. The pedal positions are controlled by defined ramps in the Manual (Pedals, Gear) interface.

Maneuver no.1 (see *Main GUI* > *Parameters* > *Maneuver* or right side of Figure 2.14) shows the manual drive-off maneuver. The clutch is set to "0" within 5 seconds. The value of the gas pedal is simultaneously set to "1" within 5 seconds. The gear is set to 1st gear.

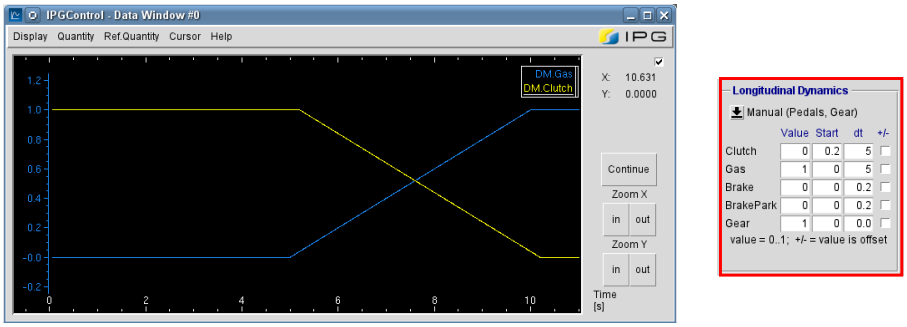


Figure 2.14: Values of DM.Gas and DM.Clutch displayed in IPGControl and the settings of the manual maneuver

FollowCourse

CarMaker provides a lateral dynamics controller named "Follow Course". It is a simple steering wheel controller which follows the center line of the road. The controller will not consider any course changes by pylons.

The controller is available in the lateral dynamics maneuver interface (see *Main GUI > Parameters > Maneuver*). If the allowable tolerance is too small, the control becomes unstable.

SpeedController

The TestRun shows the function of the simple speed controller. The main difference to the IPGDriver is that it will not consider curves. The controller can be used for tests where no high accuracy is needed in the velocity control of the vehicle. The option "Premature end when final speed is reached" has the same effect as an end condition.

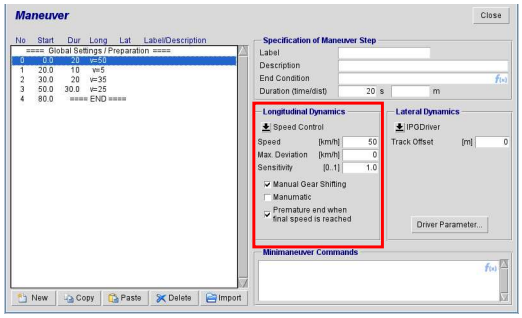


Figure 2.15: Speed Control maneuver in the longitudinal dynamics maneuver interface

For further details about the Speed Control maneuver, see the CarMaker User's Guide chapter *Longitudinal Dynamics > Speed Control*.

SteerSweep

The TestRun demonstrates the lateral dynamics maneuver "Sinus Sweep". With this maneuver, an ascending sinus steering maneuver can be performed.

The first maneuver shows a simple acceleration maneuver performed by the IPGDriver followed by a steady state part. The sinus sweep is performed in maneuver no. 2. The parameters for the sinus sweep are split into the beginning sinus and the final sinus (see Amplitude 2 and Period 2 in Figure 2.16 on the right side). The sinus steering between the beginning and the final will be interpolated.

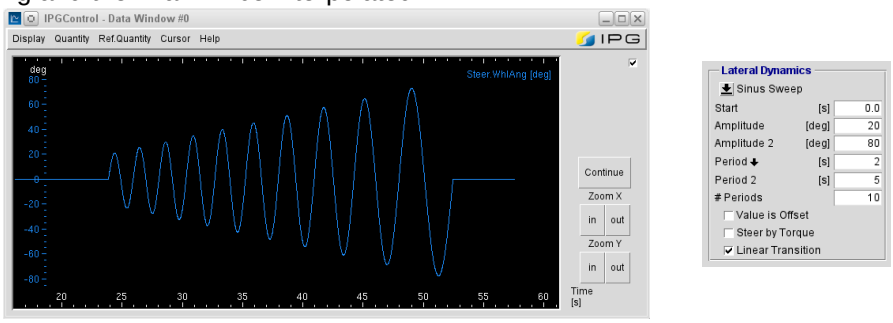


Figure 2.16: Sinus Steering sweep with 10 periods

2.4.3 Record and Replay

MeasurementReplay

In some cases, it is necessary to repeat a maneuver with the same steering and the same accelerator and brake pedal positions each time in order to evaluate the vehicle reaction. In the real world, these tests are conducted by steering robots, for instance.

The functions of this TestRun are the following:

- controlling the vehicle via an input file, defined via *Main GUI > Parameters > Input from File*
- deactivating the IPGDriver by changing the settings to manual control of the longitudinal and lateral vehicle guidance

The driver maneuver is set to manual with no inputs for this TestRun (see (1.) in Figure 2.17 "Maneuver" interface). The input variables are determined by the input file (see *Main GUI > Parameters > Input from File*). The file includes the "Steering Wheel Angle" (StWhl) and the "Gas Pedal" (Gas) as input variables which will replace the simulation variables (see (2.) in Figure 2.17). The input signal can be combined with an offset and a (conversion) factor. The filter option determines the average over the last defined (x) values (see example x =10).

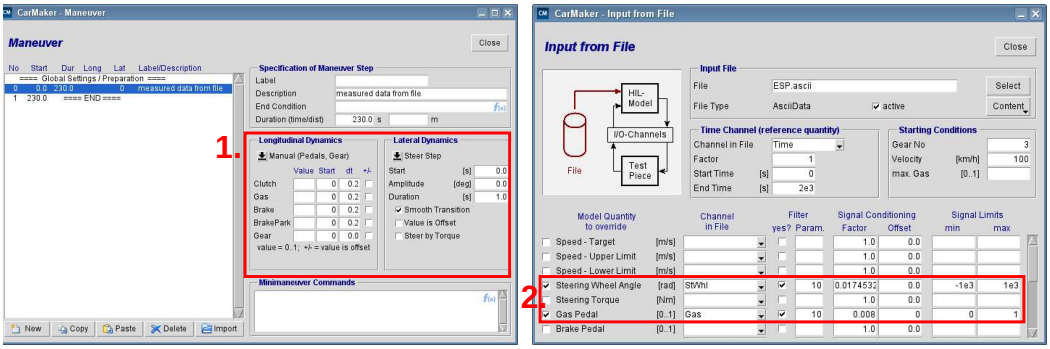


Figure 2.17: Input from file with no driver actions

MeasurementReplay_Delayed

The TestRun MeasurementReplay_Delayed demonstrates the expansion of the vehicle control by means of an input file in order to switch between the IPGDriver and the input file via minimaneuver command.

This TestRun presents the feature of

- minimaneuver commands for controlling the action from: ExtInp file: enable, disable

The input file is defined in the *Main GUI* > *Parameters* > *Input from File*. Find further information about the Input from File in the CarMaker User's Guide chapter *Input From File / Using Real World Measurements*.

The "Input from File" is automatically activated at the start of the maneuver. If you want to delay the input from file, you have to use the command

```
ExtInp file disable
```

in the first maneuver. As a result of this function, the action defined in the file is not executed until it is defined by the command

```
ExtInp file enable
```

which is shown in the minimaneuver commands of the second maneuver.

To control the function "Action from File" you can use the special commands "enable", "disable" and "restart". The command ExtInp file enable will activate the file. With ExtInp file disable, you can deactivate the input. The third command is ExtInp file restart which will restart the input from the beginning. Please see the minimaneuver command of the maneuver control via *Main GUI* > *Parameters* > *Maneuver* (Ctrl-M).

MeasurementReplay_OnOffReset

The TestRun demonstrates how to replay vehicle controlling measurements, e.g. the steering angle.

The TestRun shows the function of

- controlling the "Input from File" by minimaneuver commands
- demonstration of the maneuvers: enable, disable, restart

The input file is defined in the *Main GUI* > *Parameters* > *Input from File*. Find further information about the Input from File in the CarMaker User's Guide chapter *Input From File / Using Real World Measurements*.

In the maneuver control of the TestRun (see *Main GUI* > *Parameters* > *Maneuver*) you find different maneuvers which shows how to enable, disable or restart the input from file. To restart the file use the following command in the minimaneuver commands.

```
ExtInp file restart
```

RecordSpeedProfile

In order to ensure that the vehicle guidance remains the same each time (repeatability), the vehicle control is controlled by open-loop maneuvers. A control variable (steering angle) is predefined in order to test the vehicle reaction.

The TestRun includes the following CarMaker functions:

- saving selected CarMaker quantities via command line or manually
- converting the result file (.erg) into ASCII format (.bin) by using resutil (installation directory > bin)

To generate the drive profile, you have to save some results including "Time, Car.Gen.vx_1, DM.Steer.Ang" which must be selected in *Main GUI > Application > Output Quantities*. To save the results, you can use the DStoreSave command or the "Storage of Results" interface in *Main GUI > Storage of Results > Save* (after TestRun end). The results will be saved in the <Projectfolder> / SimOutput / <PC-name> / <Date> / SpeedInput_Hockenheim_generate.erg.

The next step is to convert the .erg file into ASCII format. Use the command prompt and go to the /bin directory of the CM installation. Type "resutil" and press enter to view all available options. To convert the .erg-file to the new file, use the command below:

```
4:  resutil -s Time,Car.Gen.vx_1,DM.Steer.Ang -om ascii -o <Path>/newfile <Path>/
    filename.erg
```

ReplaySpeedProfile

The IPGDriver can follow a "Speed Profile", which defines the velocity over time. To show this feature, the recorded and converted driving profile from the TestRun "RecordSpeedProfile" will be used.

The option is available in the longitudinal dynamics maneuver of the maneuver control interface. The profile can be restarted by activating the third tickbox. For further information see User's Guide chapter *Maneuver > Speed Profile*.

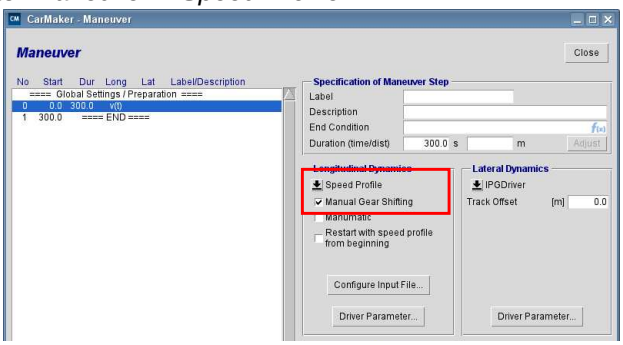


Figure 2.18: Maneuver Interface - Speed Profile

Find a comprehensive description on following a speed profile in the IPGDriver User Manual chapter *Following a Speed Profile*.

2.4.4 Special Maneuvers

SafeCleanUp

If the last maneuver in the maneuver list has a duration of zero seconds, it is called the "Final Maneuver" (see Figure 2.19). Independent of the way the TestRun is finished, stopped or canceled by an error, the maneuver will always be performed at the end of the TestRun. For further information see User's Guide chapter *Final Minimaneuver*.

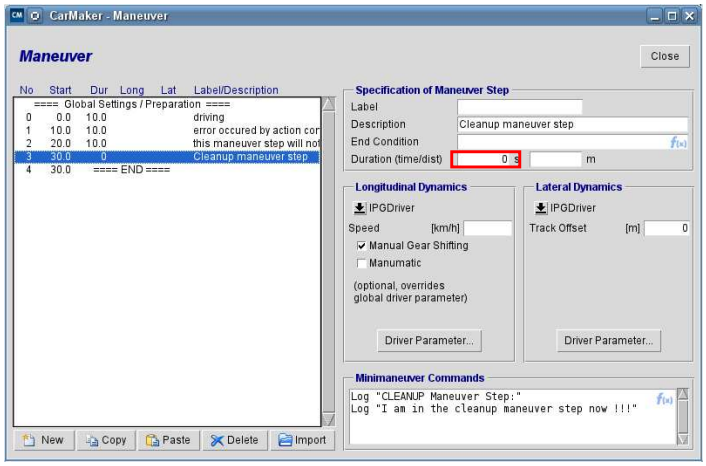


Figure 2.19: Last Maneuver definition

The maneuver can be used to reset quantities which were overwritten by DVA commands:

```
DVArel *
```

Further it can be used for user-defined post processing tasks. See User's Guide section C.1.2 'Direct Variable Access Commands' for details.

SetInitialConditions

Sometimes it is time-consuming to accelerate the car to a defined speed. For this reason, "Start Values" can be set in the Maneuver interface. Open GUI > Parameters > Maneuver and click on the maneuver "Global Settings / Preparation". The panel allows the input of velocity, gear, steer angle and track offset. There is also the option to select a ScriptControl file and to use maneuver commands. Further it can be used for user-defined post processing tasks. See User's Guide section 6.6 'Global Settings / Preparation' for details.

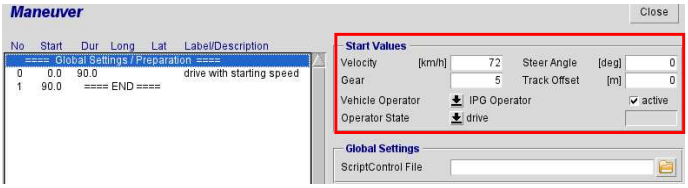


Figure 2.20: Set Starting Values

2.5 Movie

ActiveWipers

The active wipers effect serves for testing image processing algorithms. Camera systems are positioned behind the windshield for protection and a clean view. Therefore, when the wipers are active, they move through the camera's field of view. This effect should not influence the image processing and has to be managed by the algorithm. Figure 2.21 shows a snapshot of the active wipers moving through the camera's field of view.

This TestRun loads a modified version of the DemoCar vehicle which features animated wipers. For a definition of the animation, see Movie/3DObjects/Vehicles/ActiveWiper/ActiveWiper.tclgeo. The wiper is only visible when looking from the inside of the car through the windshield.

Figure 2.21: Wiper covers a part of the image

Benchmark

This TestRun includes several traffic objects. The driver controls the vehicle on the circuit. The TestRun can be used for a performance comparison of different systems.

FisheyeCamera

Automotive cameras can be distinguished by their areas of application: Front view cameras are equipped with a normal lens. They capture the close and far range in front of the vehicle and are suited for applications that detect the lane, traffic signs, objects as well as the condition of the road. For the recognition of the side and rear vehicle environment, fisheye lenses with a field of view of over 180 degrees are used. The TestRun demonstrates how to realize fisheye cameras. IPGMovie offers a special dialog to modify the camera view. The advanced camera settings are saved in the Camera.cfg file specific to the project in the Project folder > /Movie.

The lens modification options include the normal and the fisheye lens. Additional different mapping functions for the fisheye lens are available: equidistant, equal area, orthographic and stereographic. In Figure 2.22, the use of two fisheye cameras on the side mirrors is illustrated.

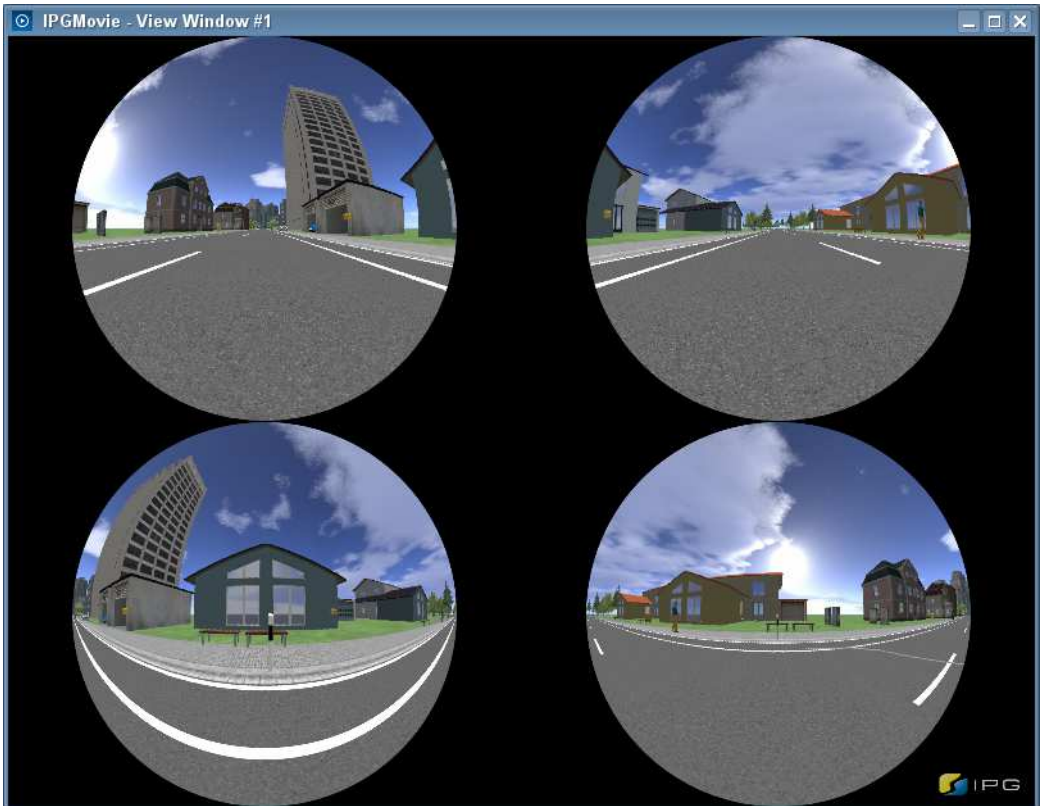


Figure 2.22: Fisheye Camera placed on side

LowHighBeam

Active lights allow for a more detailed illumination of the environment due to user-defined light sources. The lights can either be stationary and placed at a defined position, or they can move with the objects. It is possible to define street lights as well as lights on traffic objects. The lights can also be controlled to simulate adaptive headlight systems, for instance.

This TestRun demonstrates different light modes including high beam light (see Figure 2.23), low beam light, adaptive curve light and auto dimmer function. For a better visualization in IPGMovie, a dark background should be enabled (*IPGMovie > Scene > Background > Night Scene* (press key 6, active window IPGMovie)). Alternatively, the background Deep Night can be used (press key 3). For the TestRun, DemoCar_ActiveLights is used.

The description of how to define the ActiveLights can be found in the IPGMovie Manual section 4.5 'Active Lights'.

Figure 2.23: Active Lights Function

ReflectionOnWetRoad

Sometimes it is challenging for front view cameras to recognize the environment or identify it correctly in case of interferences such as reflections on a wet road. Puddles are simulated to test the robustness of image processing algorithms, for instance, under these special conditions. IPGMovie is able to visualize shadows, motion blur, puddles, blooming as well as (dynamic) reflections. The visualization of puddles on the road is integrated via mtex files (see User's Guide "Integration mtex-Files"). The reflections on the road can also be controlled via *IPGMovie > View > Quality Settings > Reflections on the road*. In order to obtain more detailed puddles, it is recommended to adjust the quality via *IPGMovie > View > Quality Settings > Quality*.

Figure 2.24: Reflection on wet road

RollingShutter

Electronic traffic signs, e.g. variable speed-limit signs on highways, may pose difficulties for automotive cameras.

In the menu Scenario / Road > Accessories Panel > Traffic sign, an electronic traffic sign is integrated as a movie object. The LED display control displays the sign section by section, imperceptible to the human eye. In a camera snapshot, however, parts of the sign are missing which need to be reconstructed from successive images. The animated sign is configured via commands in "Additional Parameters".

Figure 2.25: LED Speedlimit sign effect

Attention! In order to use this effect the quality settings in IPGMovie have to be set to "Balanced" or "Quality" and the license option "Sensor Package" has to be available. Otherwise the sign will simply be displayed and no shutter effect will be visible.Working with cameras that are attached to the vehicle requires turning off the camera smoothing via *IPGMovie > Camera > Smoothing --> off*.

2.6 Road

In CarMaker, roads can be built from segments in order to construct artificial tracks or by importing digitized roads. For the generation of a realistic environment, a range of options is available including different features such as signs, guide posts, trees, buildings, etc.

The examples include city, country and highway roads. The different TestRuns demonstrate how to include markers and infrastructure objects.

2.6.1 QuickStartGuide

The TestRuns in this folder are the solutions of the exercises in Quick Start Guide section 5.4 ’TestRun with Main Focus: Scenario Editor’ on page 62.

Step1_TestRunWithoutScenario

The exercise in Quick Start Guide section 5.4.1 ’Building a Road Network using the Scenario Editor’ starts with this TestRun.

Step2_Woods

This example shows how your TestRun should look like after following the instructions of Quick Start Guide section ’Building a Road Network’.

Step3_TestTrack

This example shows how your TestRun should look like after following the instructions of Quick Start Guide section ’Create a Closed Track with 3D Surface’.

Step4_FrictionStripe

This example shows how your TestRun should look like after following the instructions of Quick Start Guide section ’Inserting Bumps, Markers and Movie Objects’.

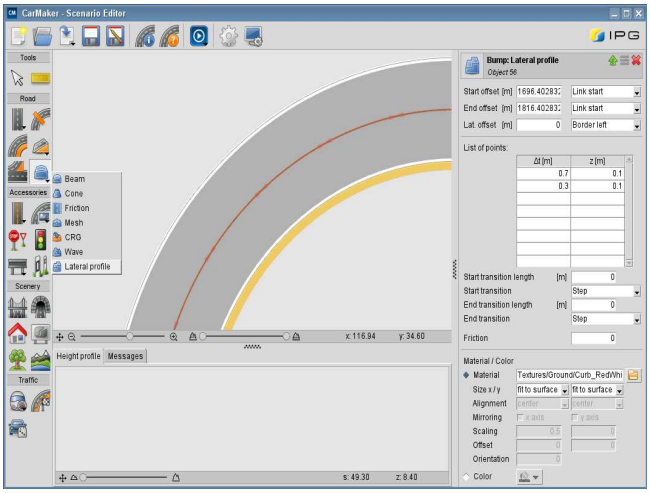
Final_Road

The exercise in Quick Start Guide section 5.4.2 ’Defining the Maneuver’ starts with this TestRun.

Final_Maneuver

This example shows how your TestRun should look like after following the instructions of Quick Start Guide section 5.4.2 ’Defining the Maneuver’.

2.6.2 Highway



ConstructionSite

Construction sites on the highway frequently interfere with the traffic flow. These circumstances demand good perception skills from both human drivers and driver assistance systems.

The TestRun presents the following function:

- multilane road with construction site

The TestRun presents a multilane highway with guardrail and yellow road markings due to a construction site. The scenario can be used for testing lane support systems as well as cruise control systems. The digitized road is inserted via *Main GUI > Parameters > Scenario / Road > Menu tab > Import road definition*. Attention! It is not recommended to edit the road file except for the route definition.

Figure 2.26: Passing construction site

Cruising_3lanes

The TestRun features a segment of a multilane freeway with an exit. Different routes can be selected for testing. The *bird's-eye view* of the road provides an overview of the routes. It can be selected by pressing and holding the Preview button.

ExitAndEnter

Building real road architectures with segments is often a very complex task. In order to include challenging road parts in the simulation, the ADAS RP interface is used to import real road sections. For more information, see CarMaker User's Guide chapter *Road: Carmaker - ADAS RP Interface*.

The CarMaker features presented in this TestRun are the following:

- importing a digitized road
- placing/moving "Traffic Objects" using the Traffic interface via *Main GUI > Parameters > Traffic* to create an interaction with the IPGDriver driven ego vehicle
- using the direction indicator via Realtime Expression in the Maneuver Control

The TestRun shows a multilane highway with an on-ramp and an exit (see Figure 2.27). A bridge crosses the highway. Via *Main GUI > Parameters > Scenario / Road > 3D Preview* (Bird's Eye View), you can preview the road. In addition, you can set the route for the ego vehicle. If you change the route, you might have to adjust the maneuver.

Figure 2.27: Highway with driveway and exit

.

2.6.3 Networks

Road networks are easily created with the third-party tool ADAS RP by HERE (see User's Guide chapter *Road: CarMaker - ADAS RP Interface*).

CountryRoad

The CarMaker features used in this TestRun are the following:

- importing a digitized road created by means of the ADAS RP interface
- using the direction indicator via Realtime Expression command, triggered by the driving distance
- using the "ManJump" command, which switches from one maneuver to another

This TestRun illustrates a road network. The ego vehicle exits a bumpy dirt road and enters a paved road. The intersection is located in the forest (see Figure 2.28.)

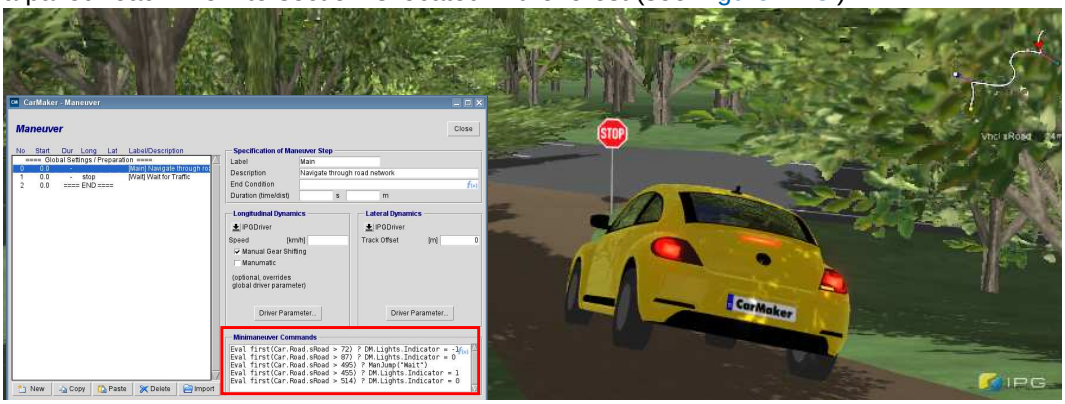


Figure 2.28: Country Roads with bumpy and bitumenized sections

The direction indicator can be set by the following command.

```
Eval first(Car.Road.sRoad > 72) ? DM.Lights.Indicator = -1
```

After reaching a maneuver driving distance of 72 m the direction indicator is set to -1 for activating the blinking on the right side (DM.Lights.Indicator = 1, for the left side; DM.Lights.Indicator = 0, for turn off)

The next command demonstrates the maneuver jump. If the condition is true, the ManJump directs to the maneuver with the label "Wait".

```
Eval first(Car.Road.sRoad > 495) ? ManJump("Wait")
```

Information on the jump function is provided in the chapter *Operators and Functions* in the User's Guide.

ParkingGarage

to be documented

Roundabouts

The TestRun illustrates road structures with more than one single road connected by roundabouts and intersections. The Bird's Eye View in Figure 2.29 provides a practical overview of the road network. There is more than one route available. With the bottom right drop-down menu you can select the different routes which are defined in the road data. Multilane roads, junctions, road markings, bridges, traffic signs, traffic lights and speed limits are supported.

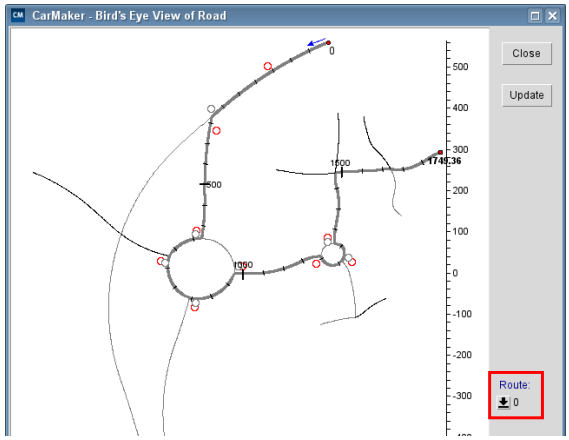


Figure 2.29: Bird's Eye View of imported network from ADAS RP

CarMaker supports the import of routes from ADAS RP into CarMaker. For detailed information, see User's Guide 'Road: CarMaker - ADAS RP Interface'.

Please note that the ADAS RP plug-ins are available with a special license option only. The ADAS RP plug-ins are included in the CarMaker installation directory. No additional installation, only a license extension is required to run the interface.

2.6.4 SampledRoads

Digitized roads can be imported in several formats. More details on the file structure can be found in the CarMaker User's Guide.

3DMapping_HigwayNo1

to be documented

3DMapping_PikesPeak

to be documented

ADASRP_Export

The TestRun presents the following function:

- importing fully digitized road sections generated via the interface Setup ADASRP

This TestRun features a route exported from ADAS RP of the urban area Stuttgart South with the starting point *Rotenwaldstraße*.

The digitized road is inserted via *Main GUI > Parameters > Scenario / Road > Road Panel > Road segment > File*.

Attention! It is not recommended to edit the road file except for the route definition. The export includes traffic signs and traffic lights.

For further information, see the CarMaker User's Guide chapter *Road: CarMaker - ADAS RP Interface*.

GPS_Track

The import of digitized roads enables the use of GPS coordinates for the routing. Find detailed information in the chapter on Digitized Roads in the CarMaker User's Guide.

KML_Export

This TestRun demonstrates the use of a route available in .kml format. The kml file can be added via the Road menu (*Main GUI > Parameters > Scenario / Road > Road Panel > Road segment > File*). For access from the GUI, the kml file should be added to the folder <Projectfolder> / Data / Road. Find more information in the chapter on *Digitized Roads* in the CarMaker User's Guide.

Nurburgring_RoadFeatures

This TestRun demonstrates several road features build into a short section (5 km) of the racing track *Nurburgring Nordschleife.*

The features of the TestRun are the following:

- including digitized roads
- include a terrain defined as geometry file (.obj) by the interface Scenario / *Road > Scenery Panel > Terrain generation*

- add local modifiers like bumps, waves, friction stripes, etc.

atlatec_KA_Suedtangente

to be documented

2.6.5 Surface

Bumps

Many factors contribute to an uneven road. The CarMaker Road includes predefined road bumps (different geometrical shapes). These can be adjusted as required and placed on the road. Detailed information is available in the User's Guide.

The TestRun shows the following functions:

- creating road bumps, e.g. speed bumps and cones
- using the "**DMjmp**" command to switch between maneuvers (e.g. subfunction for special road conditions) triggered by road markers
- road markers and the detection by the Road Sensor

To create a bump, select the predefined type that is closest to the shape you want to create. The available options are cone, beam, wave, friction, mesh, CRG and lateral profile. Each bump can be created with a user-defined color or texture. The driver is not able to detect the bump by himself, which means that you have to define a maneuver for driving over the speed bumps with adapted speed. The other solution is demonstrated in the TestRun.

The road bumps are created in the Road interface via M*ain GUI > Parameters > Scenario / Road > Road Panel > Bumps*.

The additional user-defined marker with the name "RoadAction", illustrated in IPGMovie as a small yellow sign, is created in M*ain GUI > Parameters > Scenario / Road > Traffic Panel > Markers > User-defined* (see Figure 2.30). For the beam, the marker parameter is set to 1. For the obstacle consisting of cones, the road marker parameter is set to 2. This allows for the road characteristics to be distinguished in the maneuver control.

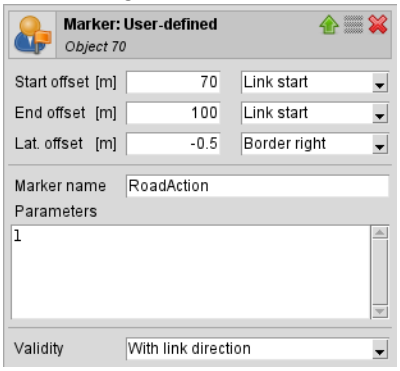


Figure 2.30: Adding user-defined road markers

For the vehicle, a "Road Sensor" for the detection of the marker "RoadAction" is defined in the vehicle settings (*Main GUI > Vehicle Data Set > Sensors > Road Sensor)*. This sensor only detects road markers that are user-defined and carry the name "RoadAction". The Road Sensors's calculated values are displayed in the quantity "Sensor.Road.D00.RMarker.Attrib.0".

Three separate maneuvers are defined in the maneuver control:

(Main maneuver) Driving as well as (submaneuvers) Speedbump and Speedcone.

```
[Sensor.Road.D00.RMarker.Attrib.0 == 2] DMjmp Speedcone
[Sensor.Road.D00.RMarker.Attrib.0 == 1] DMjmp Speedbump
```

In the main maneuver "Driving", the quantity "Sensor.Road.D00.RMarker.Attrib.0" is used as a trigger to jump to the minimaneuver with the name "Speedcone" or "Speedbump".

CRG_Patch

Real roads often have uneven surfaces which cannot be easily simulated. A special method is therefore used to measure the road segments. The result is a standardized file format (.crg) which can be integrated into the simulation.

The TestRun demonstrates the function of

- integrating real road surfaces in .crg format into the simulation (*Main GUI > Parameters > Scenario / Road > Road Panel > Bumps > CRG*)

The TestRun includes an example of how to include real road measurements via the "CRG section". First, the road section needs to be built. After this, the .crg file is included by adding a CRG section to a road segment via "*Bumps > CRG"*. When using a digital route, an additional segment has to be added to enable access to the "CRG Section". Detailed information is available in the User's Guide.

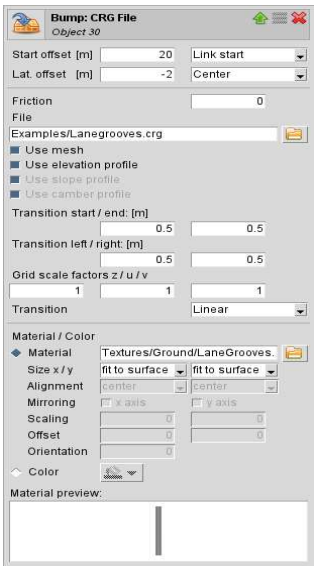


Figure 2.31: Including CRG road measurements

Curbs

Curbs can be defined in *Main GUI > Parameters > Scenario / Road > Road Panel > Bumps > Lateral profile*. The curb is defined via the dimension of the cross-section and the position relative to the road segment (start offset, end offset, lateral offset). For the color, an RGB

code or a texture file (.png) can be used. In addition, the friction coefficient can be defined. For additional information, see the User's Guide chapter *Scenario Editor > Road Panel > Bumps*.

Figure 2.32: Drivable road curbs

FrictionStripe

The friction coefficient of the road is globally defined in the standard settings. This allows for an approximation of the surface characteristics of the real road. In some cases, a local deviation of the friction coefficient is desired to replicate different conditions, for example in a µue split braking test.

This TestRun presents the function

- "Friction Stripe"

The friction stripes can be defined individually for each segment via *Main GUI > Parameters > Scenario / Road > Road Panel > Bumps > Friction*.

For more information, see CarMaker User's Guide.

RoughRoad

The TestRun demonstrates the following functions:

- creating a segment-based road
- including road bumps of the "Wave" type via *Main GUI > Parameters > Scenario / Road > Road Panel > Bumps > Wave*
- controlling the lateral position of the IPGDriver with traffic cones defined via *Main GUI > Parameters > Scenario / Road > Traffic Panel > Markers > Pylon alley*

The road shows a segment-based track which can be used for testing the driving ability of vehicles under special road conditions. The track includes different road waves as well as uphill and downhill sections with steep slopes. Traffic cones are used to direct the driver through the test track.

The waves are defined with different parameter values (see Road user interface). The descriptions of the parameters can be found in the User's Guide.

Figure 2.33: Superposition of road waves

SkiJump

In order to generate arbitrary surfaces that can be driven on, use the user interface "Mesh" via *Main GUI > Parameters > Scenario / Road > Road Panel > Bumps > Mesh.*

The TestRun demonstrates the function of

- generating drivable objects using the "Mesh" function.

The mesh is defined by a point list which includes the x-, y- and z-coordinates. The TestRun shows how to build a mesh object. Detailed information is available in the User's Guide. The "Material / Color" entry field in the mesh dialog box can be used to add a material or a color defined by RGB code, see User's Guide *Texture definition*.

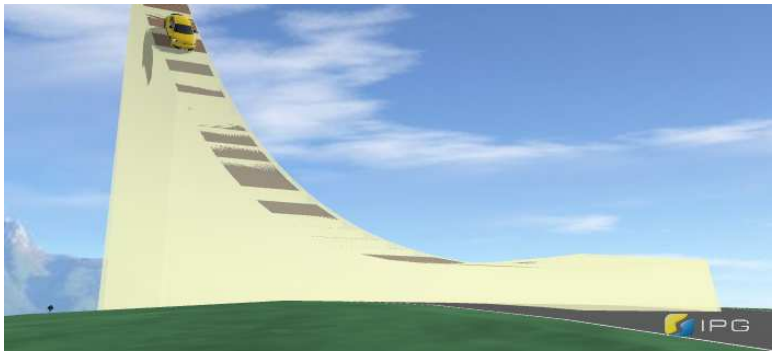


Figure 2.34: Creating drivable mesh objects

Maneuver no. 1 includes a dynamic end condition

```
DM.ManDist > 1 && Car.v < 0.1
```

in order to prevent the simulation to become stuck in this minimaneuver if the test car cannot reach the end position on the jump.

Waves

This TestRun demonstrates the creation of a wave-shaped road surface. The wave shape is characterized by the height and period.

This TestRun presents the following feature:

- Creation of a wave-shaped road surface

The Road interface provides a "Bump/Marker" for the creation of waves, accessible via *Main GUI > Parameters > Road > Segments > New Bump/Marker*. The wave modifies the flat surface of the road. The surface is created based on the wave parameters (see Figure 2.35).

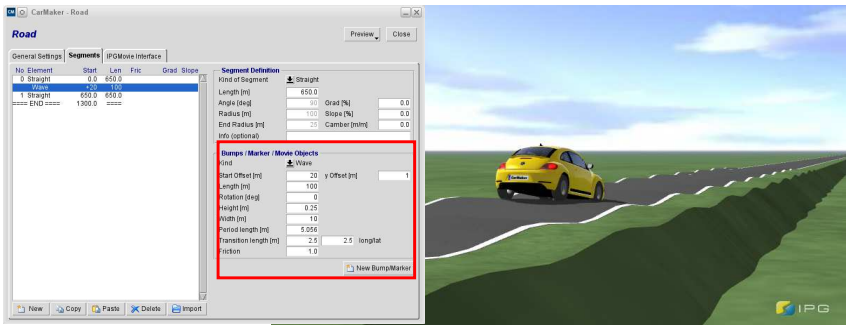


Figure 2.35: Road surface changed by bump "Wave"

For more information about the road bumps, see the User's Guide, section *Scenario Editor > Road Panel > Bumps*.

2.7 RTExpressions

DynamicEndCondition

The TestRun shows the function of the dynamic maneuver "End Conditions based on a measurement from previous maneuvers.

The functionalities used within this TestRun are:

- the creation of new "user accessible quantities"
- the definition of dynamic "End Conditions" for maneuver steps
- saving the value of quantities and calculation in realtime expressions

The TestRun FishHook shows the same content.

SpecificMeasurement

CarMaker offers the function of carrying out user-defined calculations, saving values of quantities and using a flexible trigger control in connection with the quantities in "hard" real time. A number of practical predefined functions are provided for this, which allow for a fast and clear creation of your own functions and processes. This functionality is called "Real-time Expression".

The functionality "Realtime Expressions"â has a wide range of applications. Some functions are shown in the following TestRun. (Find further information in the User's Guide section "Realtime Expressions".)

The functionalities used within this TestRun are

- the creation of new "user accessible quantities"
- the use of a built-in function which detects the change of a quantity
- the calculation of the difference of a quantity between two predefined conditions

```
Eval Qu::v_start=0
Eval Qu::Brake_Dist=0
```

The first maneuver demonstrates the creation of new quantities (v_start, Brake_Dist). In the minimaneuver command language, the RTexpr is defined by the prefix Eval. The new quantity is created with ***Qu::<name>=0***

```
Eval v_start=latch(Car.v, change(DM.Brake))
```

In maneuver no. 1, the velocity at the time of the brake actuation is saved. With the function "latch(Car.v, change(DM.Brake))", the value of the vehicle's velocity (Car.v) is stored to the variable "v_start" when the brake pedal activity changes.

```
Eval Brake_Dist=Delta2Ev(Car.Distance, change(DM.Brake), Car.v<=0.01)
```

The driven distance is saved with the variable "Brake_Dist". The distance between the actuation of the brake and standstill is recorded via the function "**Delta2Ev**". The start of measurement is triggered by the actuation of the brake pedal (change(DM.Brake)). The end of measurement is defined by the condition (Car.v<= 0.01), which stops the measurement once the vehicle velocity is less than or equals 0.01 m/s.

VariablesAndOperations

In this TestRun, the characteristic cornering speed of the vehicle is measured. The cornering speed is defined as the maximum speed when cornering.

The following functions are used:

- "Arithmetic operators"
- "Built-In functions" for selecting the minimum value coming with the installation.
- Logging function for writing to the Session Log

The vehicle accelerates and drives through the curve. While turning, the minimal speed is determined. At the end, the minimal speed is displayed.

In the maneuver control, the variable (User Accessible Quantity) Cornering_Speed is created in the CarMaker Data Dictionary. The quantity is then available in IPGControl. This is executed once the vehicle enters the curve. In the minimaneuver commands, the constant "s" can be used as an alias for the quantity "Car.Road.sRoad" ("s"=Car.Road.sRoad, e.g. "t"=Time, "pi"=π).

```
1:  Eval first(s > 499) ? Qu::Cornering_Speed=Car.v
```

The minimal cornering speed is determined via

```
2:  Eval (s > 500 && s < 570.7) ? Cornering_Speed=min(Car.v,Cornering_Speed)
```

If the driven distance is between 500 and 570.7 m, the minimal cornering speed is determined based on the quantities Car.v and Cornering_Speed. The conversion is performed in the last maneuver (Line 4). For this, the conversion factor from m/s to km/h is defined as a variable (Line 3). Once the test is completed, the cornering speed is displayed. An error log is displayed if the curve was not finished.

```
3:  Eval ms2kmh=3.6
4:  Eval Cornering_Speed= Cornering_Speed * ms2kmh
5:  Eval s>570 ? Log("Measurement Finished Cornering Speed: %.2f km/h",Cornering_Speed)
    : LogErr("Measurement not finished")
```

Test: Try to maximize the cornering speed by changing the driver parameters "Accelerations" (*Main GUI > Parameters > Driver > Standard Parameters > Accelerations, g-g diagram*).

2.8 Sensors

CollisionSensor_DetectContact

The Collision Detection module detects whether the ego vehicle body or wheels touch any traffic or road objects. For the envelope of the vehicle body, a cuboid or an extruded x-y contour can be defined. You can visualize the vehicle's "collision body" by activating the View>Show>Sensors option. For the verification of a collision in general, the side areas of the traffic object and the vehicle body are considered. Thus it can be used for crash detection or for parallel parking maneuvers. You have the option of using a cuboid for the vehicle's outer skin. The detection of collisions between the wheels and traffic objects can be activated.

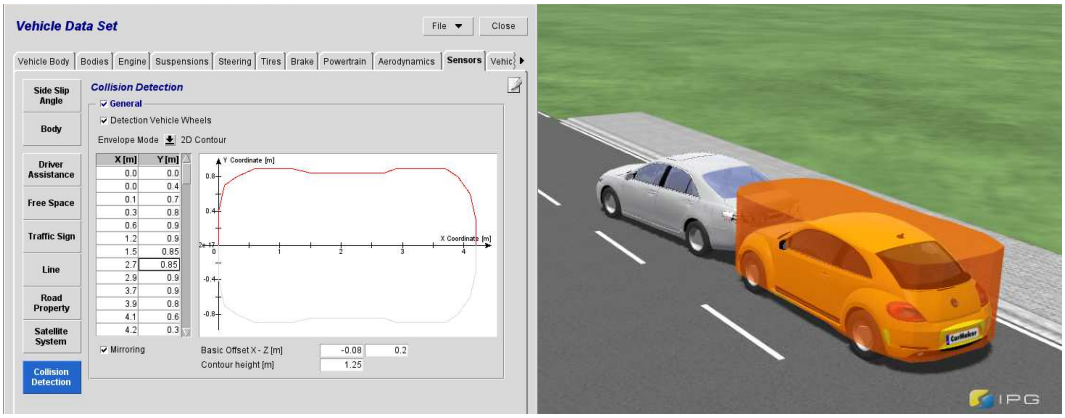


Figure 2.36: Definition of 2D-Contour for collision detection

The TestRun includes the DemoCar with a 2D contour and a target which also has a 2D contour (to show the traffic object body, click right in IPGMovie > ABRAXAS). If a collision with the vehicle body occurs, the counter **CollisionDetect.Vhcl.Fr1.Count** will not be zero.

The minimaneuver command

```
Eval first (CollisionDetect.Vhcl.Fr1.Count>0) ? LogErr ("Collision detected!")
```

will stop the TestRun via the error session log command *LogErr.* The text "Collision detected!" is printed in the Session Log (*See Main GUI > Simulation > Session Log*).

FSpaceSensor_DetectOpenSpace

Some advanced driver assistance systems (AEB, ACC) require information on the vehicle's environment. The free space can be calculated based on the sensor's measured values.

The testrun contains the following functions

- The FreeSpaceSensor which can be parameterized via *Main GUI > Parameters > Vehicle > Sensors > Free Space*
- Inner-city drives with many objects

The FreeSpaceSensor (FSSensor) indicates the distance to the nearest detected object point as well as the relative velocity of the object among other things. The User Accessible Quantities of the FSSensor are listed in the Reference Manual chapter *User Accessible Quantities - Free Space Sensor*.

The user can divide the detection range of the FSSensor into discrete solid angles. Figure 2.37 illustrates the display of the discrete cells in IPGMovie.

Figure 2.37: Free Space Sensor (IPGMovie > View > Show > Sensors)

The Free Space Sensor is configured similar to the DASensor. Range, aperture, update (Cycle) and the number of segments in horizontal and vertical direction are individually adjustable in the Free Space Sensor interface, see Figure 2.44.

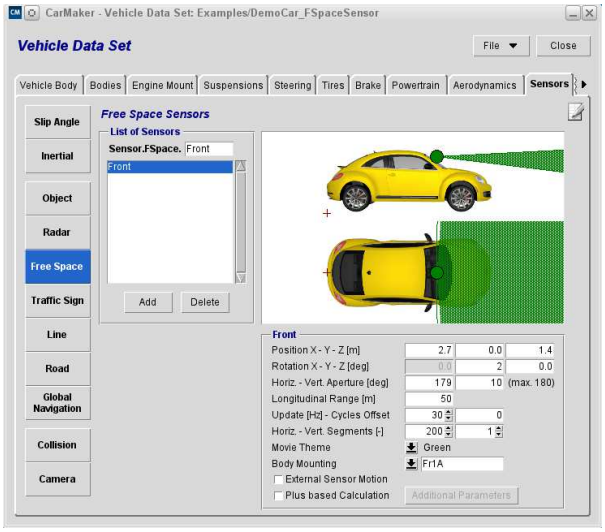


Figure 2.38: FSSensor interface in the Vehicle Data Set > Sensors

The external sensor motion can also be activated in the sensor interface. The external motion allows the relative sensor travel and rotation, which can be set with the global C variable or via Direct Variable Access on the quantities, shown in the Reference Manual chapter *User Accessible Quantities - Free Space Sensor - General*. Additionally, the movie theme of the sensor can be set to different colors (see Figure 2.37 movietheme = green) or to not visible.

FSpaceSensor_PerformanceBenchmark

The TestRun shows the simulation with the free space sensor (short FSpace). The focus is on the sensor performance in a scenario with a great amount of stationary traffic objects without name (in this case 1600 guide posts).

Figure 2.39 shows the described scenario. The top right corner of the IPGMovie window presents the overlay "Sensor view", which represent the free space detection graphically.

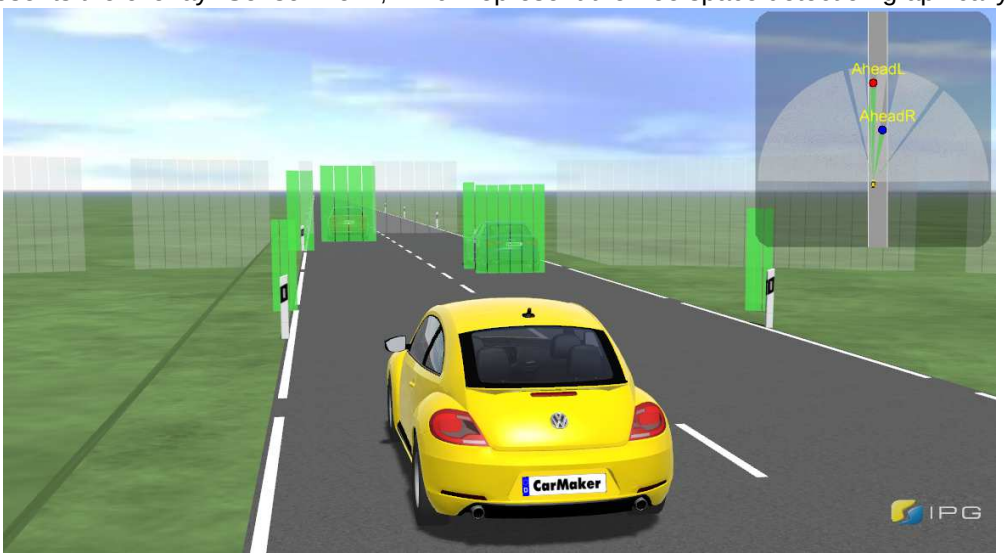


Figure 2.39: Performance test of the Free Space Sensor

FSpaceSensorPlus_DetectOpenSpace

to be documented

FSpaceSensorPlus_PerformanceBenchmark

to be documented

GNavSensor_NavigateWithSatelliteOcclusion

The Global Navigation Sensor offers the possibility of simulating the visibility of the satellites in the orbit.

This TestRun demonstrates the following:

- the functionality of the satellite system simulation
- the disturbance of the satellite signal by objects like buildings and tunnels

First the sensor needs to be activated via the Environment Parameter set. The positions of the satellites are specified by the time and date set via *Main GUI* > *Parameters* > *Environment*. See the User's Guide chapter *Environment* on how to download and integrate the file for the satellite positions.

Back in the *Main GUI* > *Parameters* > *Car* > *Sensors* > *Global Navigation* interface the parameters for the sensor can be defined. The position of the receiver, update rate and body mounting position are general sensor parameters. Further masking effects due to low elevation angle or the lack of direct view can be defined.

The satellite signal can be influenced by different errors. The error settings can be changed in the senor's *Error Parameter* interface, available in the sensor interface.

For the simulation the satellite position, a message file for the defined day is needed. Please find more information in the User's Guide chapter *Environment* about how to download the specific file.

InertialSensor_CaptureMotion

to be documented

LineSensor_DetectRoadMarkings

The information about the current driving lane and the neighboring lanes is very important for automated driving and lane support systems.

The Line Sensor module detects road markings in a specified sensor view (defined by horizontal/vertical aperture angles and range) and traffic barriers like an idealized camera. The sensor collects all lines and barriers in a sensor view and sorts them into left and right lines with ascending lateral distance to a specified frame point. The specified frame point could be the sensor origin point or any user-defined point in the vehicle frame. Additionally, the sensor provides line information, e.g. the identification code, color, type, width and height (for traffic barriers).

A detected line is divided into a number of points which can be used to calculate the lane/path prediction, for instance. Please note that the information about the relative position of these points is not provided as User Accessible Quantities by default. In fact, there are many different possibilities to define the distance of the vehicle to a line or a spline through the line points. The global header file LineSensor.h can be used to read the coordinates of the line points and implement your own algorithm using the CarMaker C code interface. For more information, see the User's Guide chapter *Line Sensor*.

Figure 2.40: Line detection with LineSensor (settings: IPGMovie>View>Show>Line Detection)

ObjectSensor_DetectTrafficContour

In some cases the traffic object's contour should be detected with a higher resolution. The Object Sensor provides the output of the nearest point of the object in the sensor area.

The TestRun demonstrates the following:

- the distance measurement to traffic objects with a 2D-contour
- the difference between the *Calculation Class* of the *Nearest Point* and *Reference Point*

The Object Sensor can be placed anywhere on the ego car. The TestRun includes an ego car equipped with multiple Object Sensors for the simulation of parking sensors. The Object Sensor's calculation class is parameterized to detect the nearest point in the sensor area. It will then measure the distance to the nearest point of the traffic object, even if the object is defined with a 2D contour. Figure 2.41 shows the IPGControl window on the left. The selected quantity in the IPGControl window shows the distance to the nearest point. The visualization is shown in IPGMovie on the right side.

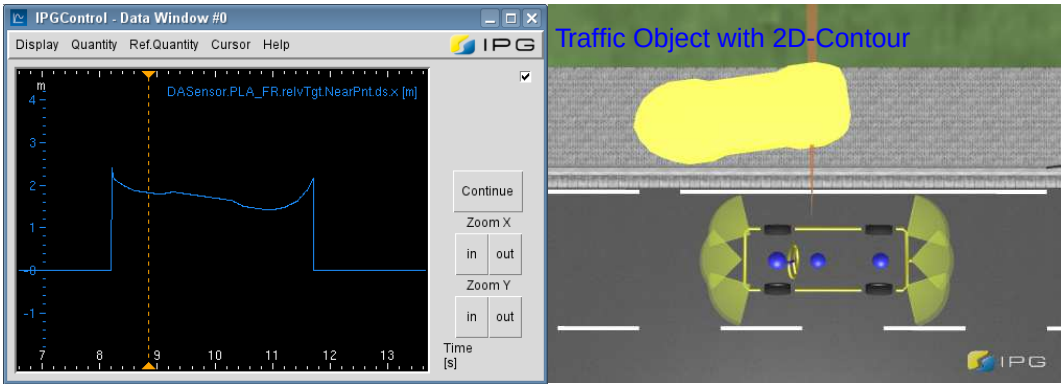


Figure 2.41: DASensor Contour Detection of traffic object

The calculation class for the Object Sensor can be selected in the sensor interface. Other detection points of the target vehicle are the reference point (located in the center of the rear-most surface).

ObjectSensor_SelectTargets

to be documented

RadarRSI_Motorway

The Radar Raw Signal Interface determines the propagation of electromagnetic waves in the virtual scene. It delivers the channel impulse response, sufficient information to reconstruct the analog signal. For further information please refer to the User's Guide and Reference Manual.

RadarSensor_MergeObjects

The Radar (HiFi) Sensor detects targets incorrectly when they are to close to each other (grouping of objects).

RadarSensor_Occlusion

Occlusion effect: Pedestrian is not detected, when occluded by the bus. The car in front of the bus is detected with a weakened signal.

RadarSensor_FalsePositives

The Radar (HiFi) Sensor detects vehicles in its perception area. It shows phenomena like occlusion, grouping, noise, latency, False Positives, etc.

RoadSensor_DetectBendAndMarker

The Road Sensor allows the detection of specific road characteristics such as the curvature, slope, lane width and number as well as user-defined markers (see the Reference Manual chapter *Road Property Sensor*). The use of the Sensor and the interface to the output quantities is presented below.

The features provided within this TestRun are the following:

- the definition of a Road Sensor
- detection of user-defined road markers

Several attributes like the lane width, curvature, global position of the preview point, number of lanes and road marker attributes (speed limit) or the longitudinal and lateral slope are detected.

This data can be important for various applications such as lane keeping assist systems, lane departure warning, autonomous driving, sign detection, energy management, prescanning, optimization of the fuel consumption or detection of wheel lifting. The sensor model has different operating modes for the detection of specific road information such as the deviation from the lane center line or the current lane width. The third mode is used for the detection of speed limits or user-defined markers.

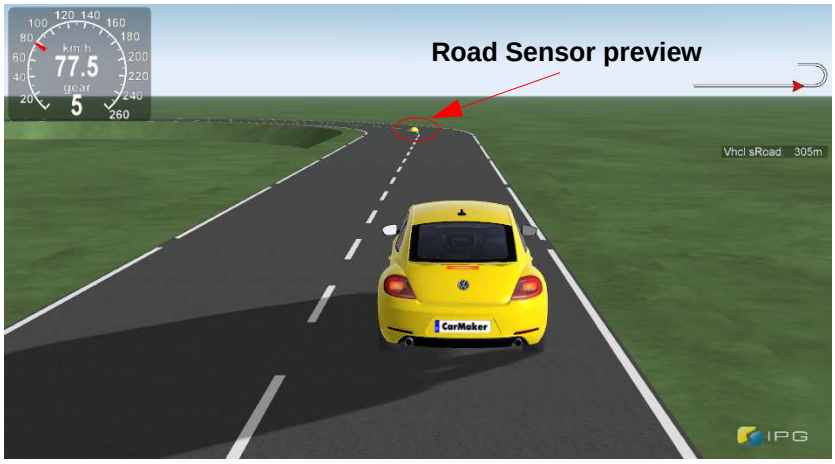


Figure 2.42: Road Sensor preview distance

The TestRun includes the DemoCar_RoadSensor which is equipped with two Road Sensors: one for speed limit detection and one for road property detection. Figure 2.42 shows the sensor preview, shown by the yellow dot on the road.

You can visualize the sensor measurements in the IPGControl application, see Figure 2.43. For example you can see the cars velocity and vehicle control (VC.Brake) dependent on the road sensor measurements.

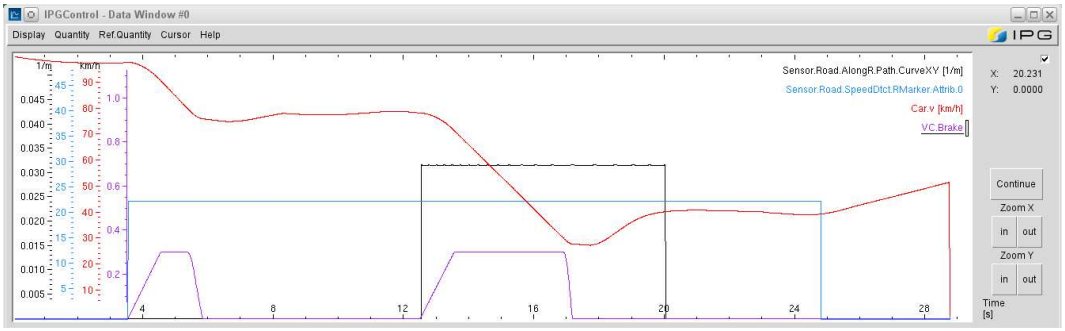


Figure 2.43: IPGControl window with measurements of the TestRun

The measured values of the sensors are included in maneuver control. The first maneuver is executed until the Sensor *SpeedDtct* detects a speed limit (see End Condition Man. no. 0). The vehicle decelerates via manual pedal control until the velocity is less than the detected speed limit, defined by the End Condition of maneuver no. 1

```
Car.v < (Sensor.Road.SpeedDtct.RMarker.Attrib.0)
```

Now the IPGDriver will take control of the vehicle until the curvature of the road is detected by the second road sensor named *AlongR.* The end condition of maneuver no. 2 shows the detection of the curvature

```
Sensor.Road.AlongR.Path.CurveXY !=0
```

Now the manual braking is performed until the velocity is below 30 km/h. The driver will take control of the car until the end of the road.

For further information, see the Reference Manual chapter *Road Sensor*

SAngleSensor_CaptureSideSlip

The Slip Angle Sensor and the Inertial Sensor provide measured values of the current driving condition of the vehicle.

The TestRun demonstrates the following:

- using the *Slip Angle Sensors*
- using the *Inertial Sensors*

At least one Slip Angle Sensor has to be defined. One Slip Angle Sensor is defined at the vehicles center of gravity by default. Additional sensors can be placed on the vehicle body. If the vehicle body is defined as a flexible body, the respective part (Frame Fr1A or Frame Fr1B) can be selected. The sensor output can be displayed in IPGControl as the quantity *Sensor.SAngle.<Name>.Ang*. Find further information in the User's Guide chapter *Slip Angle Sensor*.

The Inertial Sensor represents an inertial measurement unit which reports the acceleration, velocity, position and rotational velocity as well as acceleration. The sensor can be placed anywhere on the vehicle. The parameters of the sensor can be defined in the user interface including the mounting position and rotation as well as the frame to be used as the mounting frame. In *Calculation Class*, the reference for the measurement can be set. The references "*global frame"* only or the "*global frame plus body"* (in-/ excluding the gravitation) can be set to calculate values for both references. For further details, see User's Guide chapter *Intertial Sensors* or the description of the quantities in the Reference Manual chapter *User Accessible Quantities*.

TSignSensor_DetectTrafficSigns

The Traffic Sign Sensor provides information about passed traffic signs. The important sensor parameters are mounting position, aperture and the range, for instance. Due to this, the sensor's detection capability is similar to a real camera. However, there is no implementation of an image recognition function.

The TestRun demonstrates the following:

- the functions of the Traffic Sign Sensor
- the selection of detectable signs.

The sign detection can be set to detect all signs or only selected signs, see Figure 2.44 on the right side (limited to 10 signs).

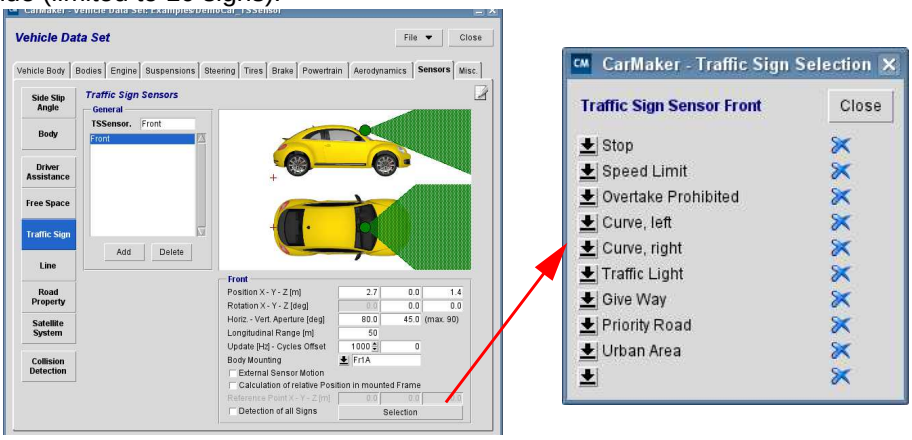


Figure 2.44: Traffic Sign Sensor

The information about the sign detection is provided by quantities for each sign, including the detection state, values and further information. For a detailed description, see the Reference Manual chapter *Traffic Sign Sensor*.

The sign has to be within the detection range of the sensor in order to be detected. In addition, the front of the sign needs to face in the direction of the ego vehicle.

USonicRSI_Parking

The Ultrasonic Raw Signal Interface determines the propagation of sound pressure waves in the virtual scene. Enhanced by the transducers directivity and a time dependend threshold a complete sensor is modeled. For further information please refer to the User's Guide and Reference Manual.

2.9 TestAutomation

2.9.1 ScriptControl

Straight_TrailerSwingingDVA

This TestRun is controlled via the ScriptControl file "TrailerSwingingDVA.tcl" located in "<Installation Directory>/Data/Script".

The features of the TestRun are the following:

- start TestRuns via the ScriptControl file
- DVA mechanism via the ScriptControl file, which allows you to directly manipulate quantities states during the simulation

To open the ScriptControl go to *Main GUI* > *Simulation* > *ScriptControl.* The ScriptControl interface shows a script overview on the top window and the Console at the lower window. Then open the ScriptControl file "TrailerSwingingDVA.tcl" via the "Open" button, see Figure 2.45 (1.). You can also create new files or edit loaded files by using the other buttons.

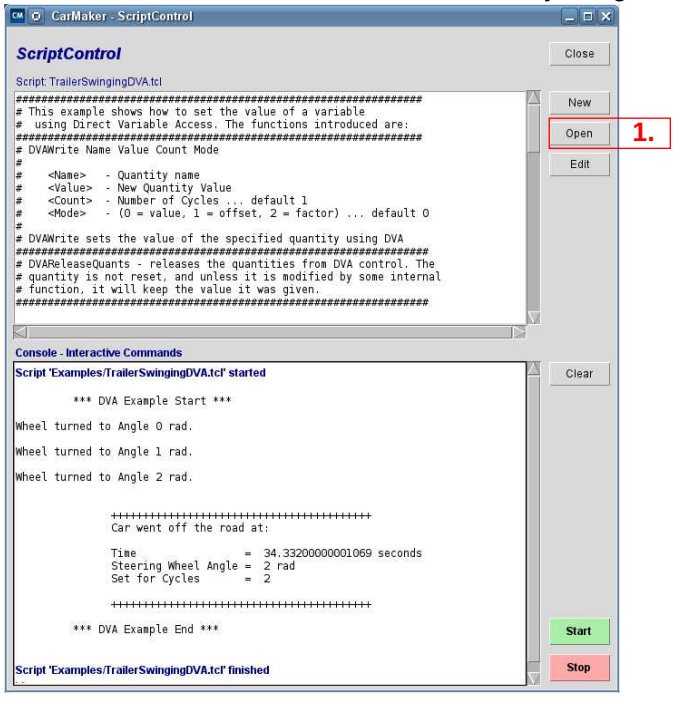


Figure 2.45: ScriptControl user interface

All used commands in the ScriptControl file are listed in the CarMaker Programmer's Guide chapter *ScriptControl Command Reference.*

Further information about the ScriptControl tool ScriptControl are available in the CarMaker Programmers Guide chapter *ScriptControl.*

2.9.2 TestManager

FailSafeAndDiagnostics.ts

This Test Series demonstrates the usage of a script file (.tcl) to automatically call user-defined procedures before and/or after a Test Run.

Its contents are:

- Usage of a script file (.tcl) to call procedures
- FailSafe Test (CarMaker/HIL)
- TestRun variations using Test Manager

The script file is embedded into the Test Series using the CarMaker parameter "*ScriptFile*". The parameter "*StartProc*" determines the procedure to be called before a TestRun. Analogously the parameter "*EndProc*" defines the procedure to be called after a TestRun is completed, see Figure 2.46.

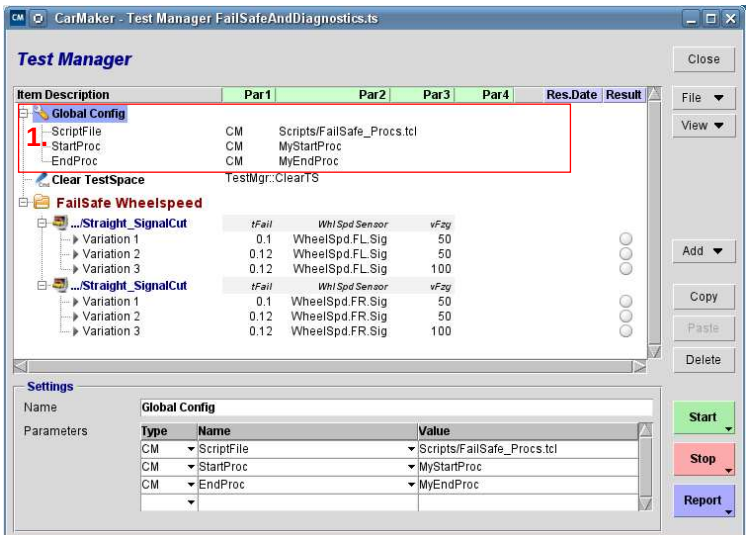


Figure 2.46: Including script files in the Test Manager

The statement in the second line of the StartProc (here called MyStartProc) retrieves the context in which the procedure was called. This could be executing the StartProc at the beginning of a TestSeries, a Group, a TestRunGroup or a TestRun. If the variable $key is equal to "TestRun" the statements in line 4 to 7 are executed.

```
 1:  proc MyStartProc {key name args} {
 2:    switch $key {
 3:     TestRun {
 4:           PowerOn
 5:           Diag connect
 6:           Diag clearerrors
 7:           Diag disconnect
 8:         }
 9:    }
10:  }
```

After a TestRun has finished the procedure *"MyEndProc" (see code below)* will be called. Here, possible errors will be read from the ABS/ESC ECU (CarMaker/HIL) as long as the TestRun has been executed successfully so far, see line 10.

Lines 12 to 16 read out the error state of the ECU. If there are any errors (see line 18) the TestRun is marked as "bad". Finally, the number of errors and the error codes are written to TestSpace variables.

```
 1:   proc MyEndProc {key name args} {
 2:    global TestNo
 3:
 4:    switch $key {
 5:     TestSeries  {}
 6:     Group    {}
 7:     TestRunGroup {}
 8:     TestRun {
 9:           # only evaluate criteria if simulation was successful so far
10:        if {[TestMgr::GetResult] != "good"} return
11:
12:           PowerOn
13:           Diag connect
14:           set Diag geterrors
15:           set Errors [Diag geterrorcodes]
16:           Diag disconnect
17:
18:           if {[llength $Errors] > 0} {
19:            TestMgr::SetResult bad
20:            foreach err $Errors { Log $err }
21:           }
22:           set ::TS::Test($TestNo,nErrors) [llength $Errors]
23:           set ::TS::Test($TestNo,ErrCodes) $Errors
24:           incr TestNo
25:         }
26:    }
27:  }
```

In the TestSeries three NamedValues are varied. The TestRun **Straight_SignalCut** uses these NamedValues as variables. In the Minimaneuver Commands the signal defined by the variable *WhlSpdSensor* is cut for the duration of *tFail* (see code below). Additionally, both variables will be written to the Session Log.

```
1:  Log "WhlSpdSensor: $WhlSpdSensor"
2:  Log "    tFail: $tFail s"
3:  FSTChCt $WhlSpdSensor
4:  [DM.ManTime>=$tFail] FSTChCn $WhlSpdSensor
```

The vehicle speed is varied in the Test Manager using the NamedValue *vFzg*.

This test series is intended to run on CarMaker/HIL with an ABS/ESP system connected. ECU diagnostics has to be configured for the actual ECU.

KPIsAndDiagrams.ts

In this test series the braking test is executed with a varying brake pedal actuation.

The test series shows the functions:

- variation of TestRun parameters
- creation of diagrams in the Test Manager

First the TestRun Braking is added to the test series. With the function *Add > Characteristic)* a Realtime Expression is defined for calculating the distance travelled by the car during the braking maneuver.

If the actual maneuver number is greater than zero the distance between the start of the braking maneuver and the point when the vehicles velocity becomes smaller than 0.01 is saved in the quantity BrakeDist.

```
1:  (first () ? Qu::BrakeDist=0); DM.ManNo>0 ? BrakeDist=Delta2Ev(Car.Road.sRoad,
    change(DM.Brake), Car.v <=0.01)
```

The test is passed successfully once the distance is not larger than 14m. Depending on the result of the criteria, the lights turn red (failed) or yellow (Warning) or green (passed), seeFigure 2.47

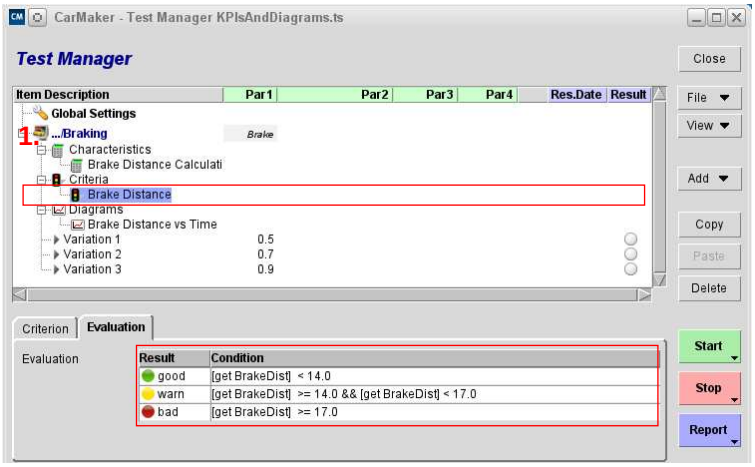


Figure 2.47: TestRun criteria in the Test Manager

The settings of the diagram are the lables of the axes and the content. In the content you have to insert the names of the quantities which are to be used in the diagram.

Further the variation of the brake pedal value is set as variation. The NamedValue is defined in the TestRun **Braking** (double click on the TestRun in the Test Manager to load it in the CarMaker GUI), in the "Braking" maneuver.

TestSpaceVariables.ts

This Test Series demonstrates the usage of TestSpace variables.

TestSpace variables can be defined and used in a Test Series. In this example, they are defined in the Global Settings and used to set several NamedValues, see Figure 2.48. TestSpace variables can be accessed using this syntax: $TS::NameOfVariable.

The TestSpace (*TestManager GUI > View > TestSpace)* gives an overview of all TestSpace variables.

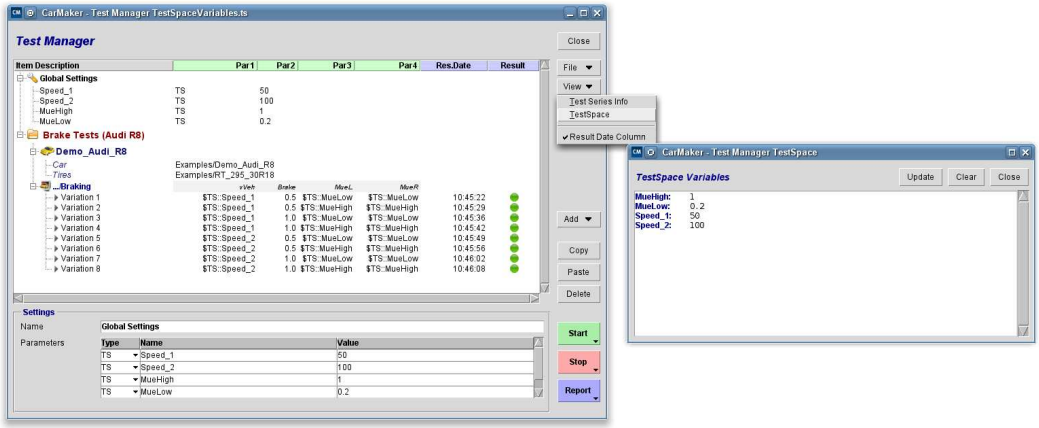


Figure 2.48: Including script files in the Test Manager

Further information can be found in the CarMaker User's Guide's section *TestSpace Variables*.

VariationsAndSkipping.ts

In this test series, the slalom test is driven with increasing speed. The criteria checks after each variation, whether the vehicle has hit any pylon or not. In case a pylon was touched, all remaining variations are skipped thanks to a script extension.

The features of this example are:

- variation of TestRun parameters
- skipping the variation by user defined condition

The path to the script file is set to the CarMaker variable (Type: CM) with the name Script File, see Figure 2.49 (1.). Also the function (*MyEndProc*) which is called after the TestRun is finished.

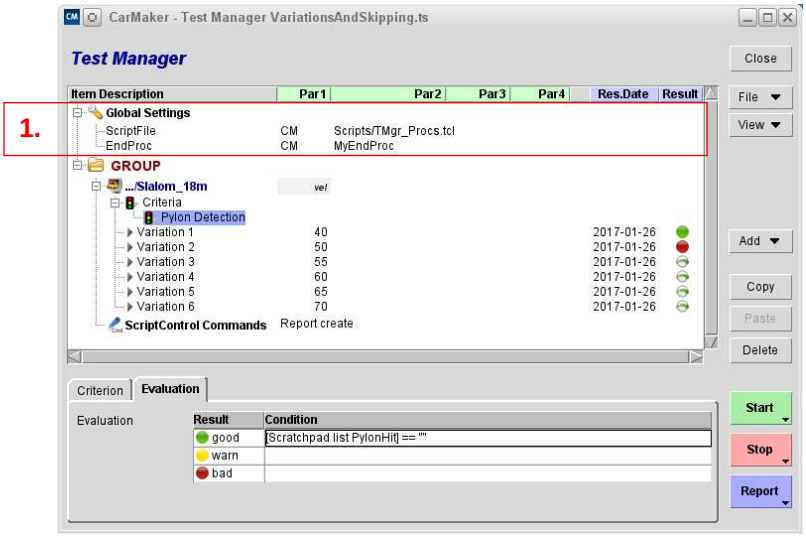


Figure 2.49: Including script files in the Test Manager

In the script file "TMgr_Procs.tcl" (see *<Installation directory >/Data/TestRun/Examples/BasicFunctions/TestAutomation/Test Manager/Scripts/TMgr_Procs.tcl*) the flow of the program is controlled by the switch/case function. The end proc is called after the end of every Test Manager iteration. The program shows some Log function which prints the defined output to the ScriptControl console (see *Main GUI > Simulation > ScriptControl*).

The end process of the TestRun shows the skip function. If the TestRun result differs from "good", the function `::TestMgr::SkipToEnd TestRunGroup` jumps to the end of the group.

```
 1:  Log "Reading file [info script]"
 2:
 3:  proc MyStartProc {key name args} {
 4:    switch $key {
 5:      TestSeries   {Log "MyStartProc TestSeries:  $name"}
 6:      Group      {Log "MyStartProc Group:    $name"}
 7:      TestRunGroup  {Log "MyStartProc TestRunGroup: $name"}
 8:      TestRun     {Log "MyStartProc TestRun:   $name"}
 9:      default     {Log "MyStartProc unknown key '$key'"; TestMgr::Stop}
10:    }
11:  }
12:
13:
14:  proc MyEndProc {key name args} {
15:    switch $key {
16:      TestSeries   {Log "MyEndProc TestSeries:  $name"}
17:      Group      {Log "MyEndProc Group:    $name"}
```

```
18:      TestRunGroup  {Log "MyEndProc TestRunGroup: $name"}
19:      TestRun     {
20:        Log "MyEndProc TestRun:   $name"
21:        # if criteria false skip the remaining test if this group
22:        if {[TestMgr::GetResult] != "good"} {
23:          ::TestMgr::SkipToEnd TestRunGroup
24:        }
25:      }
26:      default     {Log "MyEndProc unknown key '$key'"; TestMgr::Stop}
27:    }
28:  }
```

With the command at the end of the test series "Report create" a summary of the test series is created and saved to the "SimOutput" folder in the actual project directory.

VarySimulinkParameters.ts

This test series shows the variation of Simulink workspace variables. The variable is defined as gain parameter in the CarMaker Simulink model **BodyCtrl** (see *CarMaker > Vehicle > CarAndTrailer > Forces > External Suspension Forces).* The variation of the parameter is set as NValue "*BodyCtrl_k*".

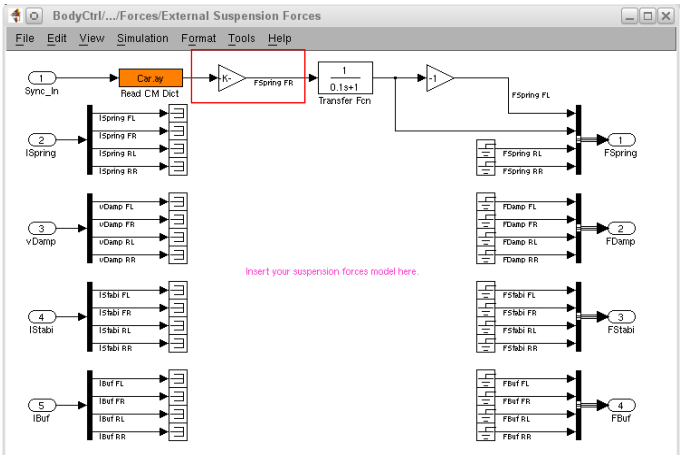


Figure 2.50: Location of parameter *BodyCtrl_k* in the Simulink model

CarMaker for Simulink is needed for this example. Without CarMaker for Simulink and the correct model "BodyCtrl" the variation will have no effect on the simulation!

VehicleConfiguration.ts

This TestSeries demonstrates the selection of a test vehicle in the Test Manager. If no vehicle is specified in the Test Manager, the vehicle defined in the TestRun will be used.

The test series shows:

- selection of the vehicle and tires in the Test Manager

A vehicle can be added to a TestSeries in the Test Manager by choosing *Add > Vehicle.* This creates a new entry in the Test Series (see Figure 2.51). After this, the vehicle as well as tires, trailer or additional loads can be selected.

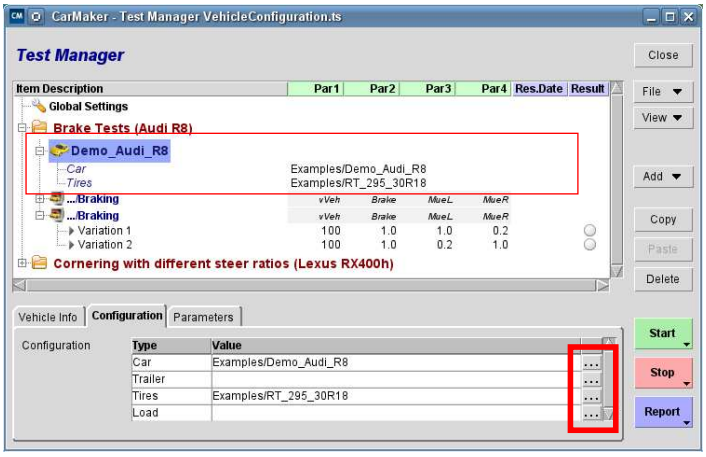


Figure 2.51: Vehicle configuration in the Test Manager

Additionally, this TestSeries demonstrates parameter variations to the TestRun *Braking.* The parameters to be varied are: vehicle speed (vVeh), brake pedal position (Brake) and the friction coefficients on the left and the right side of the road (MueL, MueR). These parameters are defined as NamedValues (NV) and are used in the TestRun (see Figure 2.52).

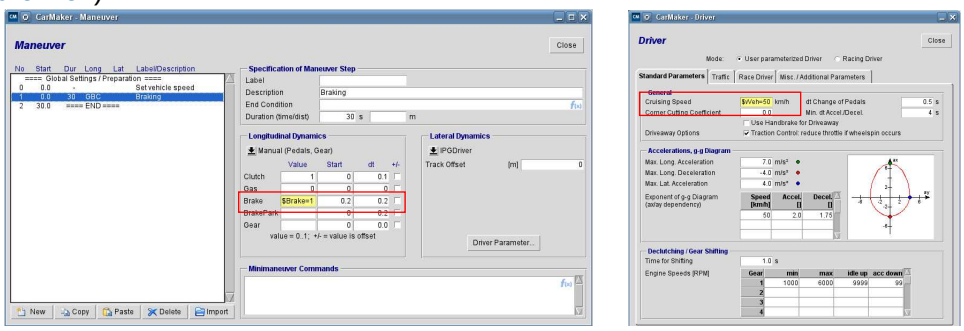


Figure 2.52: NamedValues in the maneuvers and driver settings

SlalomTest.tws

to be documented

2.10 Traffic

Basics_2D_Contours

Sensor objects (objects defined in the Traffic GUI) can be recognized by the *Object Sensor* and the *Free Space Sensor*. The standard setting yields objects that are represented by a rectangular box. For more detailed applications, the shape of the object can be adjusted.

This TestRun presents

- the activation of the traffic objects 2D contour in the Traffic user interface

The 2D contour can be defined by activating the tick box for 2D contour and clicking on the yellow polygon (see (1.) in Figure 2.53). The "2D Contour Park" will appear. After setting the contour, the object will no longer be represented by a squared shape when viewing it from the top. Instead, it will have the user-defined contour. This contour can be used for a more realistic detection of the distance with sensors. More information is provided in the User's Guide chapter *General Attributes of Traffic Objects*.

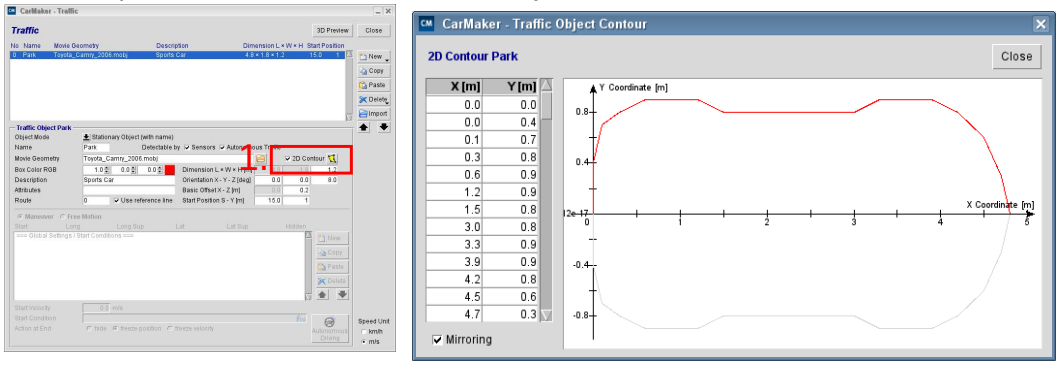


Figure 2.53: 2D Contour definition for traffic objects

Basics_Animals

to be documented

Basics_Ball

to be documented

Basics_MovingObjects

In order to move the traffic objects, you have to set the "Start Velocity" in the Traffic interface for the traffic object (*Main GUI > Parameters >Traffic*). The traffic object will not recognize any signs or other traffic objects. It will drive with the defined "Start Velocity". The "Speed Unit" can be selected in the bottom right corner of the Traffic user interface.

Basics_People

to be documented

Basics_StaticObjects

Objects can be placed using the traffic interface (see Figure 2.54) by clicking on (1.) "New" (press and hold for predefined objects). The object mode (2.) defines "Movable Objects". For these objects, new quantities are created (see IPGControl Traffic). For "Stationary Objects (with name)", less quantities will be created. With the option 'Without Name', no quantity will be created

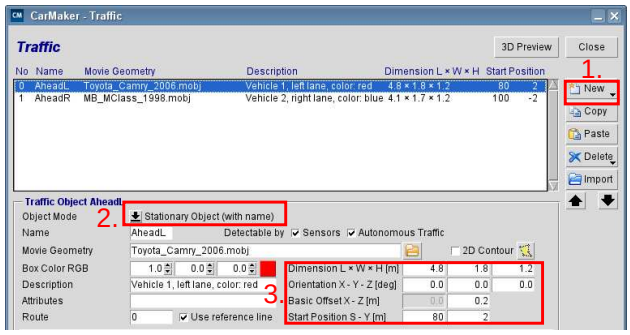


Figure 2.54: Traffic interface placing objects

The object dimension is set for the rectangular box. The box is shown in IPGMovie via the Abraxas mode (right click and select). The starting position is defined relative to the global coordinate system (Fr0).

Man_Autonomous

to be documented

Man_DynamicRouteChange

to be documented

Man_LaneChangeCompare

to be documented

Man_LaneChangeCompare.ts

to be documented

Man_Path_PedestrianCrossing

to be documented

Man_Path_PullOut

to be documented

Man_Path_UTurn

to be documented

Man_UseRecords

to be documented

OncomingObjects

The TestRun demonstrates the features:

- define oncoming traffic objects
- setting the route for the traffic objects

The TestRun demonstrates several overtaking maneuvers by traffic objects and the ego car. The traffic objects drive along the selected route. When creating oncoming traffic, the route (in this case route 1) that is opposite to the route that the ego car is driving on (route 0) should be selected (see Figure 2.55).

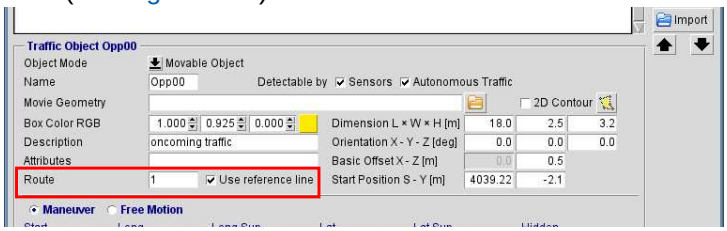


Figure 2.55: Defining the route for the traffic objects in the *Main Gui* > *Parameters* > *Traffic*

RoadTriggeredObjects

The maneuver control of the traffic objects is based on time or distance and can be parameterized with absolute values or in incremental steps.

The TestRun shows the following functions:

- using the special road markers "DrivMan Cmd" in the Road interface via *Main GUI* > *Parameters* > *Road* > *Segments* > *Bumper/Marker*
- setting the traffic maneuver to "Time" (incremental time) and "Abs. Time" (absolute time)

An offset for the traffic time can be done by setting the quantity **Traffic.TimeOffset** via DVA command.

```
DVAwr Traffic.TimeOffset Abs 1 0
```

If *Traffic.Freeze* is set to 1, the objects remain frozen in their current position.

```
DVAwr Traffic.TimeOffset Abs 1 0
```

The DVA commands in the TestRun are connected to the road markers "DrivMan Cmd" which are defined in the Road interface via *Main GUI* > *Parameters* > *Road* > *Segments* > *Bumper/ Marker*. Once the ego vehicle reaches the road marker, the command is executed and sets the Traffic.TimeOffset to the defined value via DVAwr.

2.11 VehicleModel

2.11.1 Trailer

CarMaker includes the *Trailer Editor* for modeling user-defined trailers. The subsections of the trailer model are similar to the car model including the trailer body, bodies, suspension, tires, sensors, hitch, brake, aerodynamics and an option of visualizing the trailer with a geometry model. A full description of the subsections is provided in the User's Guide chapter *Trailer Model*.

LoadsAndSideWind

The reaction of the trailer is affected by loads, wind and other external influences. The loads are connected with the trailer's body.

The features including in this TestRun are:

- adding loads to the trailer by *Main GUI > Load*
- driving through side wind areas

The loads are defined in the *Load* interface accessible via the *Main GUI*. By defining a negative X value, the load is added to the trailer, if one is attached to the car. Figure 2.56 shows the car-trailer combination attacked by side wind. The wheel loads on the left side show a higher normal force than the wheels on the right side. This is caused by the side wind and the unbalanced trailer load.

Figure 2.56: Car-Trailer-combination is attacked by side wind (picture modified)

The trailers aero marker is specified in the *Main GUI > Parameters > Trailer > Bodies > Areo Marker.* How the wind force is applied to the trailer is defined in the trailer interface, see Figure 2.56. For example the option "*Step*" applies the force in one simulation cycle as soon as the trailer's *Aero Marker* passes the *Wind Marker*. The option*"Ramp"* will built up the wind force in a linear function, beginning once the *Aero Marker* of the trailer passes the *Wind Marker* and ends as soon as the rearmost point of the trailer passes the marker. Then the force is fully built up.

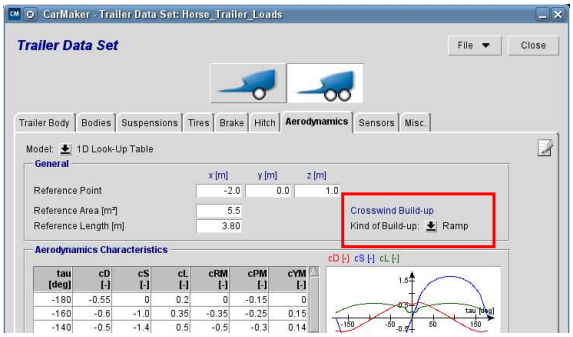


Figure 2.57: Trailer crosswind Built-up

Another load function is the trim load in *Main GUI > Parameters > Car/Trailer > Bodies*. In the check of the static state of the car/trailer, the trim loads are also considered. These are saved in the vehicle/trailer model. The loads, in contrast, are only considered in the simulation of the TestRun. These have no influence on the static state check of the vehicle or trailer.

For more information on the loads and trim loads see User's Guide chapter *Loads and Trim Loads*.

SingleAxleTrailer

The TestRun includes a uniaxial trailer. The position of the hitch can be defined using the Bodies interface of the car. The trailer is built with a sleeve axle which is characterized by a straight translation of the wheel carrier (see the Reference Manual chapter *Sleeve Axle*). The brake model includes an overrun brake (with friction), a hydraulic brake or a user-defined model. The editor also provides a crank axle, semi-trailing arm axle, linear 1-DOF, linear 2-DOF and includes an external file.

TwoAxleHorseTrailer

The TestRun includes a two-axle trailer. You can add the second axle by activating the two-axle semi-trailer mode shown in Figure 2.58. Here you can also see the Bodies interface for the definition of the positions of the wheel carrier and the wheels, trim loads and aero marker.

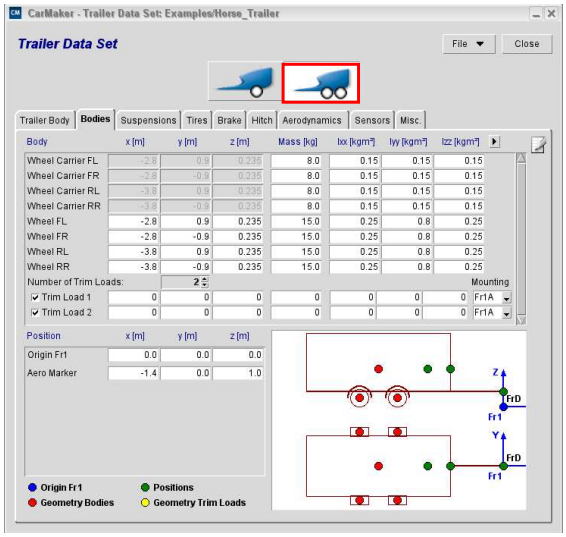


Figure 2.58: Trailer Editor, switch between uniaxel and two-axle trailer

Chapter 3
Driver Assistance

This section shows how to use CarMaker in various "Driver Assistance" applications for simulation and testing. The TestRuns will focus on different driver assistance applications for example Lane Support Systems and Emergency Braking Assist.

3.1 QuickStartGuide

The TestRuns in this folder are the solutions of the exercises in Quick Start Guide section 5.2 ’TestRun with Main Focus: ADAS’ on page 32.

Step1_CreateRoadAndManeuver

This example shows how your TestRun should look like after following the instructions of Quick Start Guide section 5.2.1 ’Building a Road Network Using the Scenario Editor’ and section 5.2.2 ’Defining a Maneuver’.

Step2_AddTraffic

This example shows how your TestRun should look like after following the instructions of Quick Start Guide section 5.2.3 ’Adding Traffic’.

Step3_AddCollisionDetection

This example shows how your TestRun should look like after following the instructions of Quick Start Guide section 5.2.4 ’Collision Detection’.

Step4_SetupAEBSystem

This example shows how your TestRun should look like after following the instructions of Quick Start Guide section 5.2.5 ’Setting up an AEB System’.

Step5_SetupNamedValues

This example shows how your TestRun should look like after following the instructions of Quick Start Guide section 5.2.6 ’Using NamedValues’.

Step6_VariationInTestManager.ts

This example shows how your TestSeries should look like after following the instructions of Quick Start Guide section 5.2.7 'Executing Multiple TestRun Variations using the TestManager'.

3.2 EuroNCAP

The European New Car Assessment Program (EuroNCAP) is a vehicle safety rating program for testing vehicles in comparative cross-company tests. New tests including active safety systems for pedestrian protection will rate the accident prevention of the vehicles systems

AEB_VRU_TestSeries.ts

The Test Series includes different runs and scripts for automatically executing and analyzing the TestRuns. The series shows the AEB VRU (Autonomous Emergency Braking Vulnerable Road User) test case in three different implementations.

Scenario **CVFA** (Car-to-VRU Farside Adult): Vehicle travels forwards with constant velocity towards an adult pedestrian. The pedestrian is crossing the street from the farside. If no braking is applied the pedestrian hits the vehicle at 50% of the vehicles width.

Scenario **CVNA**(Car-to-VRU Nearside Adult): Vehicle travels forwards with constant velocity towards an adult pedestrian. The pedestrian is crossing the street from the nearside. If no braking is applied the pedestrian hits the vehicle at 50% of the vehicles width.

Scenario **CVNC** (Car-to-VRU Nearside Child): Vehicle travels forwards with constant velocity towards a child. The child is coming out of hiding behind a parking car to cross the street from the nearside. If no braking is applied the child hits the vehicle at 50% of the vehicles width.

The Runs will be executed with a variation of the vehicle speed. Also the impact position of the VRU will be varied in CVNA-25 an CVNA-75 to change the impact position to 25% and 75% of the vehicles width. The Test Series shows a useful summary of the different test modifications. The Figure 3.1 shows a part of the CVNC scenario. The settings shows the impact position, velocity of the VRU, AEB switched on, FCW (Forward Collision Warning) on. The single runs are analyzed whether a collision has occurred.

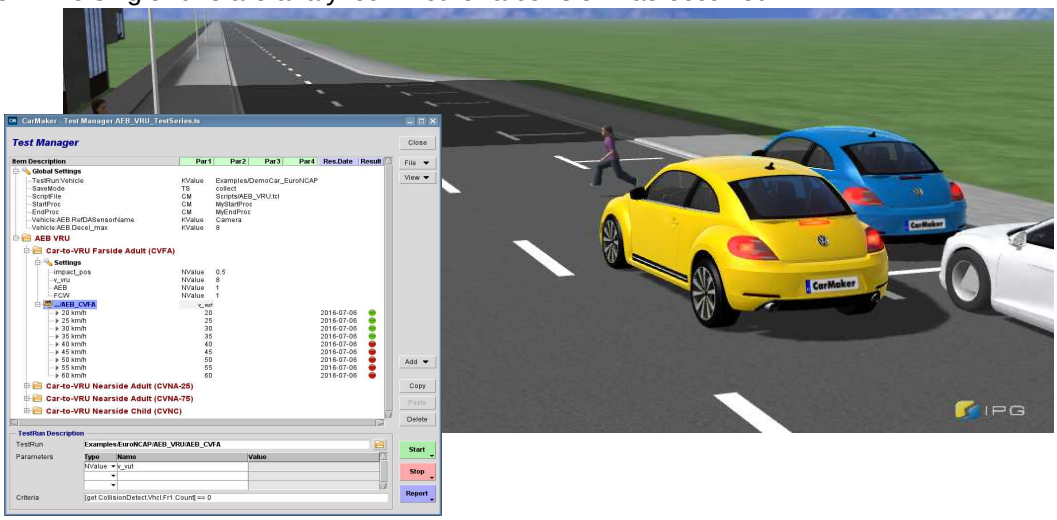


Figure 3.1: AEB VRU TestSeries with Test Manager

The following section will describe the usage of the TestSeries "AEB_VRU_TestSeries.ts".

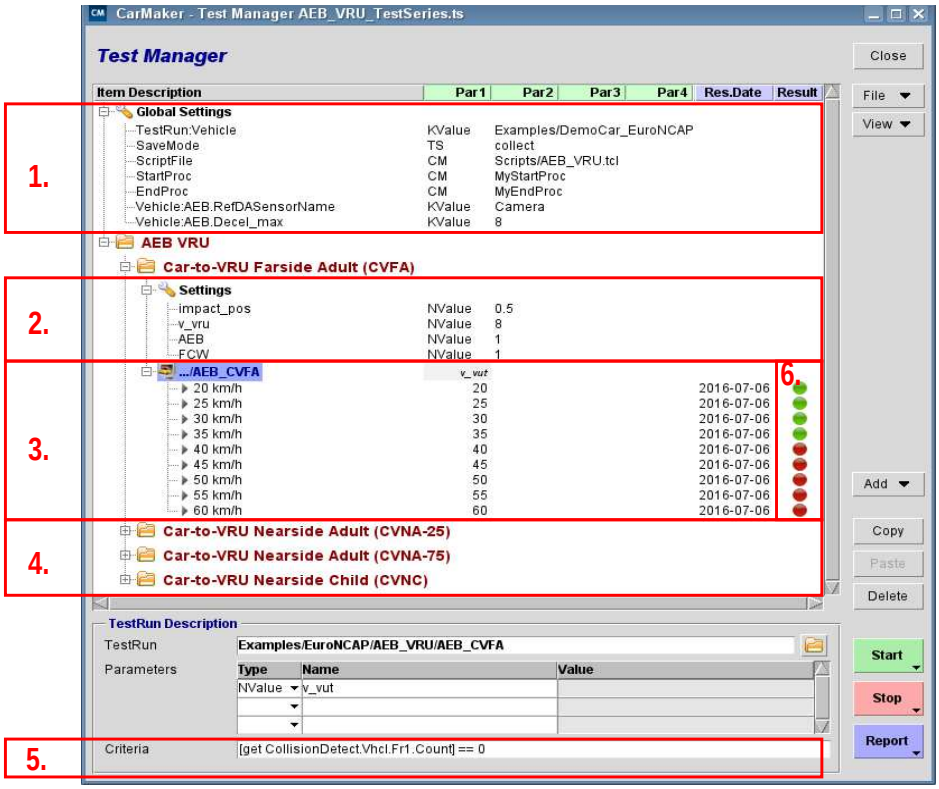


Figure 3.2: TestManager AEB_VRU_TestSeries

The the first section of the TestSeries, global settings are applied, see 1. in Figure 3.2. The settings applied in this case are listed below:

- Changing the vehicle to "DemoCar_EuroNCAP"
- Apply the "SaveMode" "collect"
- Declaration of the location of the ScriptFile (.tcl) and the procedures used as StartProc and EndProc ("MyStartProc" and "MyEndProc")
- The KeyValue "*Vehicle:AEB.RefDASensorName*" is set to the name of the DASensor used for the AEB application
- The maximal deceleration of the AEB system is set using the KeyValue "*AEB.Decel_max*"

In the next section, see 2. in Figure 3.2, the settings for the first TestRun to be varied are fixed:

- *"impact_pos"* defines the impact position of the VRU if no braking is applied (0.5 = vehicle center)
- "v_vru" defines the crossing speed of the VRU
- "AEB" defines the activation of the AEB function (1=active)
- "FCR" defines the activation Forward Collision Warning (1=active)

The third section (3. in Figure 3.2) contains the TestRun used as well as variations of the variable "v_vut". In these variations, the ego-vehicle's speed is increased in discrete steps.

The remaining part of the TestSeries, see 4. in Figure 3.2, contains other TestRuns and demonstrates variations of the impact position of the VRU and of the crossing speed.

The following section will explain the Minimaneuver Commands in the maneuver definition by the example of the TestRun "AEB_CVNC". The TestRun is designed for vary the vehicle velocity and adapt the timing for the VRU crossing the road.

The first section (Line 1-12) describes the initialization and declaration of CarMaker variables:

```
29:  Eval v_ped = ($v_vru=5)/3.6
30:  Eval v_vut = ($v_vut=40/3.6)
31:  Eval impact_pos = ($impact_pos=0.5)
32:  Eval y0_ped = -5.0
33:  Eval ped_length = 0.4
34:  Eval ped_width = 0.4
35:  Eval veh_width = 1.7
36:  Eval d_ped_acc = 0.5
37:  Eval a_ped= v_ped**2/(2*d_ped_acc)
38:  Eval t_ped_acc = 2*d_ped_acc/v_ped
39:  Eval d_ped_run = 3-ped_width/2+veh_width*(impact_pos - 1/2)
40:  Eval t_ped_run = d_ped_run/v_ped
```

Line 13 triggers the start of the VRU by the vehicles driving distance

```
41:  Eval d_veh_trigger = (t_ped_acc+t_ped_run)*v_vut
```

`Line 14 sets the variable t0 = 0 after each maneuver start` (performed once after condition is true)

```
42:  Eval first() ? t0 = 0
```

Line 15 sets the DM.ManTime = t0 when the vehicle is closer to VRU than d_veh_trigger, (performed once after condition is true)

```
43:  Eval first( ((Traffic.P00.sRoad-ped_length/2)-(Vhcl.sRoad+4.05-3.28)) <
     d_veh_trigger) ? t0 = DM.ManTime
```

Line 16 is performed each cycle for the calculation and setting of the VRU position when the VRU accelerate to moving speed

```
44:    Eval ((t0>0) && (DM.ManTime < t0 + t_ped_acc))  ? Traffic.P00.ty = y0_ped +
     v_ped**2/(4*d_ped_acc)*(DM.ManTime-t0)**2
```

Line 17 is performed each cycle for the calculation and setting of the VRU position when the VRU is moving across the street

```
45:    Eval ((t0>0) && (DM.ManTime >= t0 + t_ped_acc)) ? Traffic.P00.ty = y0_ped +
     d_ped_acc + v_ped*(DM.ManTime-t0-t_ped_acc)
```

TestRun "End Condition"

```
CollisionDetect.Vhcl.Fr1.Count > 0 || Traffic.P00.ty < -4
```

The TestRun ends after the end condition is met. Either a collision is detected so the collision count will exceed zero or the VRU has finished crossing the street (Traffic.P00.ty < -4).

Box 5. in Figure 3.2 shows the criterion used to evaluate if a TestRun was successful or not:

```
[get CollisionDetect.Vhcl.Fr1.Count] == 0
```

The TestRun will evaluate as successful ("good" / green light) if no collision occurred. Everytime a collision happens, the value Collision.Detect.Vhcl.Fr1.Coun is increased resulting in the TestRun to be evaluated as "bad" (red light). To use the collision detection it has to be activated in the vehicle data set *(Vehicle Data Set > Sensors > Collision Detection)*.

AEB_Interurban.ts

This TestSeries demonstrates a possibility how to test an AEB (Autonomous Emergency Braking) system. In the TestSeries the three TestRuns listed below are varied:

- **AEB_CCRs**

- **AEB_CCRm**
- **AEB_CCRb**

The TestRun AEB_CCRs (Car-to-Car Rear Stationary) shows the ego-vehicle driving onto a stationary vehicle. The speed of the ego-vehicle (vehicle under test / vut) is varied between 30 kph and 80 kph, see Figure 3.3 (1.) NamedValue "v_vut".

As long as no collision occurred the TestRun is evaluated as successful (green light). If one ore more collision occurs the TestRun is not successful (red light). This condition is implemented using the statement below, see Figure 3.3 (4.):

```
[get CollisionDetect.Vhcl.Fr1.Count] == 0
```

The second TestRun AEB_CCRm (Car-to-Car Rear Moving) is very similar to the TestRun AEB_CCRs. The main difference is that the target vehicle is driving with a fixed velocity (default: 20 kph). The speed of the ego-vehicle (vehicle under test) and the speed of the target vehicle are modified using the NamedValues (parameters) "v_vut" and "v_target", respectively.

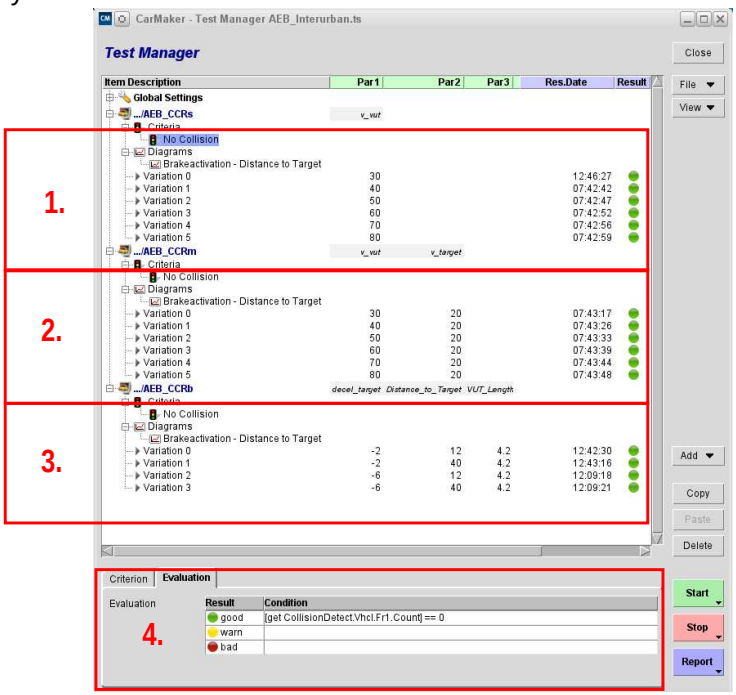


Figure 3.3: AEB TestSeries for interurban braking scenario

In the TestRun AEB_CCRb (Car-to-Car Rear Braking) the ego-vehicle initially follows the target vehicle with a fixed distance of 12m or 40m. If a set of tolerances regarding the vehicles' velocities and distance is met the target vehicle begins to decelerate with a certain deceleration (-2m/s^2 or -6m/s^2).

This behavior is accomplished using a set of maneuvers. In maneuver no.0 the TestRun is initialized. In maneuver no. 1 the ego vehicle's velocity as well as the distance between the ego-vehicle and the target vehicle are evaluated:

```
1:  Eval (abs(Car.v-50) >1 && abs(DASensor.FrontRadar.relvTgt.NearPnt.ds-
    $Distance_to_Target=12) < 0.5) ? ManJump("Follow")
```

If the vehicle's velocity is within a range of +-1kph of the desired speed and the desired distance is met with a tolerance of +- 0.5m the maneuver "Follow" is started. If the distance is too large the maneuver "Shorten_dist" is started (line 2) and if it is to small the maneuver "Increase_dist" is started (line 3).

```
2:  Eval ((DASensor.FrontRadar.relvTgt.NearPnt.ds-$Distance_to_Target=12) > 0.5) ?
    ManJump("Shorten_dist")
```

```
3:  Eval ((DASensor.FrontRadar.relvTgt.NearPnt.ds-$Distance_to_Target=12) < 0.5) ?
    ManJump("Increase_dist")
```

In the maneuver "Shorten_dist" the ego-vehicle's speed is increased for a short period of time to decrease the distance to the target vehicle. Analogously, the speed is decreased in the maneuver "Increase_dist". In either of the above maneuvers the maneuver is changed to "Follow" if the respective conditions are met. If they are not met after 30 seconds the TestRun is cancelled, see line 2.

```
1:  Eval (abs(DASensor.FrontRadar.relvTgt.NearPnt.ds-$Distance_to_Target=12) < 0.5) ?
    ManJump("Follow")
2:  Eval first(DM.ManTime>30) ? LogErr("Test condition not reached! Shortening the
    distance was not successful!")
```

In maneuver no.4 the quantity Qu::T01_BrakeTrigger is set to "1". Thus, the target vehicle starts its braking maneuver, see End Condition of traffic object T01's first maneuver (1. in Figure 3.4). The target vehicle decelerates with the deceleration specified by the Named-Value decel_target, see Figure 3.4 (2.).

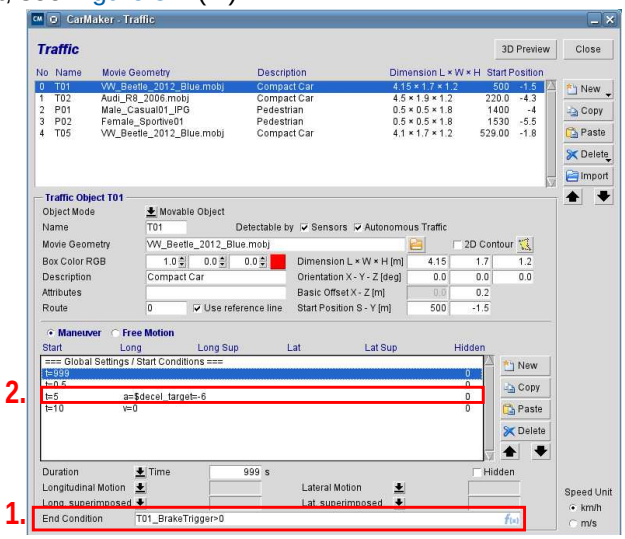


Figure 3.4: Maneuver of the traffic object for TestRun AEB_CCRb

When the AEB system starts braking the maneuver is changed from maneuver 4 to 5. The TestRun ends if the ego vehicle's velocity has fallen below 0.01 m/s, see end condition of maneuver 5.

Chapter 4
Powertrain

4.1 DrivingCycles

DrivingCycles.ts

The TestSeries and TestRun used in this example includes the functionality of

- redriving an "advanced speed profile" with the targets of speed, gear, upper and lower speed limit
- the fuel consumption map and
- the TestManager for including pre- and post processing functions regarding the actual setup of the TestRun and the test results.

In the following, a short instruction is given as to show how to use the Driving Cycle simulations implemented in CarMaker. It is assumed that the users have already had some basic knowledge in using CarMaker and know about the file structure of CarMaker. When this is not the case, we recommend you at first to have a look at the 'Quick Start Guide' of CarMaker.

Open the TestSeries "DrivingCycles.ts" file in Test Manager. Here, the vehicle can be selected within the "Vehicle Selection and Parameter Variator" settings. This can be done by varying the KValue and giving the address of the Vehicle Data Set of the corresponding vehicle. Some other parameters can also be set here when needed. Then, the simulation is started by pressing the button 'Start' in the Test Manager GUI. If this is done, all the cycles included in the Driving Cycles Test Series will be simulated. If a particular cycle needs to be simulated, the user should right click on the particular Driving Cycle TestRun or the particular variation and select the command "run selected".

In this chapter we highlight the regulations to speed tolerance of each driving cycle, depending on which the success or failure of simulating a particular driving cycle will be judged. These criteria are implemented in CarMaker by using the real-time expressions. For example, in the driving cycle WLTC the following regulations to the speed tolerance are defined:

Speed tolerances greater than prescribed shall be accepted, provided the tolerances are never exceeded for more than one second on any occasion and no more than ten such deviations per test.

This prescription for WLTC will be implemented in CarMaker MiniManeuver as below (as to the interpretation of the real time expressions, please refer to the CarMaker User's Guide chapter Realtime Expressions):

```
 1:  Eval first() ? Qu::Violations=0
 2:  Eval first() ? Qu::Viol_exceed=0
 3:  Eval first() ? Qu::Violations_Prop=0
 4:  Eval first() ? Qu::count=0
 5:  Eval first() ? Qu::count_exceed=0
 6:  Eval first() ? Qu::diffvio=0
 7:  Eval first() ? Qu::diffvio_exc=0
 8:
 9:  Eval (Car.v<DM.v.Trgt_LimitLo || Car.v>DM.v.Trgt_LimitHi) ? Violations=Violations+1
10:  Eval diffvio=diff(Violations,Time)
11:  Eval change(diffvio) ? count=count+1
12:  Eval (diffvio>0 && hist(diffvio,1000)>0) ? Viol_exceed=Viol_exceed+1
13:  Eval diffvio_exc=diff(Viol_exceed,Time)
14:  Eval change(diffvio_exc) ? count_exceed=count_exceed+1
15:  Eval (change(diffvio_exc) && count_exceed%2 != 0) ? LogWarn("The speed tolerances
     are exceeded for more than one second !")
```

The simulation of the driving cycle WLTC is then successful, when count<=10 and count_exceed<=0. These two criteria will be prompted in the TestManager under the WLTC TestRun, which is also illustrated in Figure 4.1. The result of the simulation is then indicated by the status lamps in the Test Manager. A comparison of the simulated velocity with target speed, upper speed limit and lower speed limit for every driving cycle can also be illustrated. Figure 4.2 shows an example. As the simulations are finished, a report can be easily produced by clicking on "Report" in TestManager.

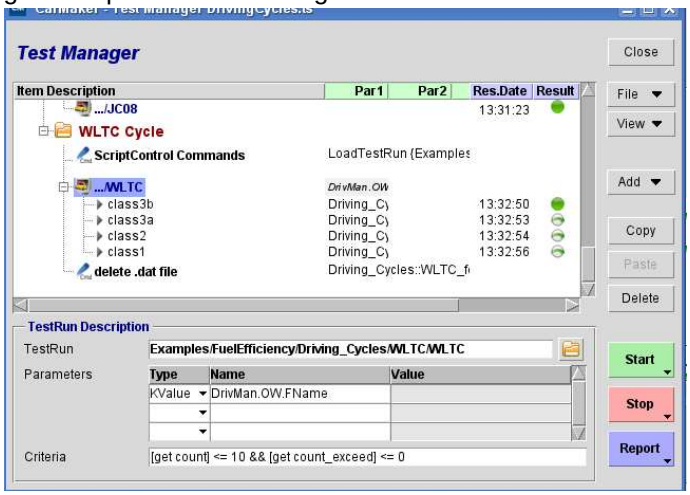


Figure 4.1: WLTC TestManager

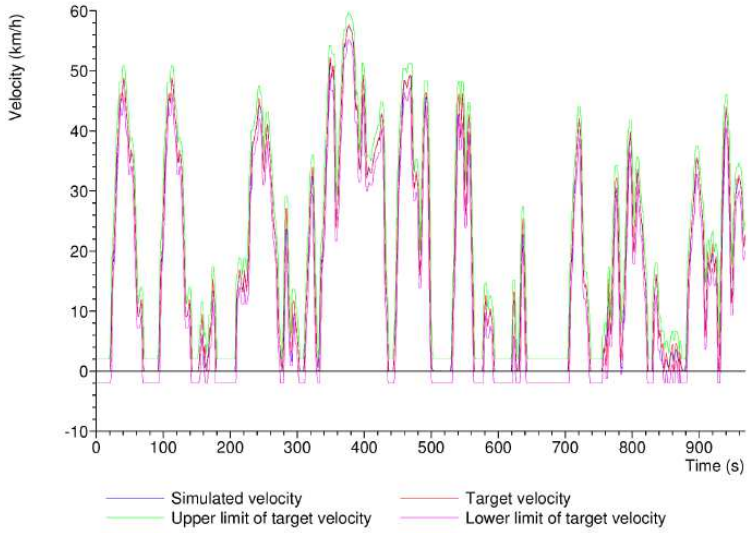


Figure 4.2: Comparison of simulated velocity with target and limits in Test Manager implementation

In addition, the fuel consumption calculations can be done for the particular cycle. For this, the user has to parameterize the "fuel consumption" dialogue box in the Vehicle Data Set > Powertrain > Engine > fuel consumption. A simple real-time expression can be written in command box of a second mini maneuver with duration of zero seconds to log the fuel consumption and the simulation results again in session log (see the example below for results evaluation of the WLTC cycle).

```
16:  Eval count=count/2
17:  Eval count_exceed=count_exceed/2
18:  Eval first() ? Log("Fuel consumption = %.3f l/100km.",PT.Control.Consump.Fuel.Avg)
19:  Eval first(Violations==0) ? Log("There is no speed out of range during the whole
     cycle.")
20:  Eval first(Violations!=0) ? Log("Sum of time in which the actual speed out of range
     = %.2f s.",Violations/1000)
21:  Eval first(Violations!=0) ? Violations_Prop=Violations/(1000*Time)
22:  Eval first(Violations!=0) ? Log("Violation proportion during the whole cycle = %.2f
     %.",100*Violations_Prop)
23:  Eval first(count_exceed!=0) ? Log("There are %d times that the speed out of range,
     and among them there are %d times that \nlast for more than one second
     !",count,count_exceed)
24:  DVArel *
```

As an example, the results of the WLTC driving cycle with DemoCar are illustrated in Figure 4.3, which shows an average fuel consumption of 6.615 l/100km and no violations during the whole cycle is occurred. In contrast, the DemoCar runs slightly out of the speed range in ARTEMIS Motorway 150 cycle. As illustrated in Figure 4.4, at the time when the vehicle runs out of tolerance, a warning message will pop up.

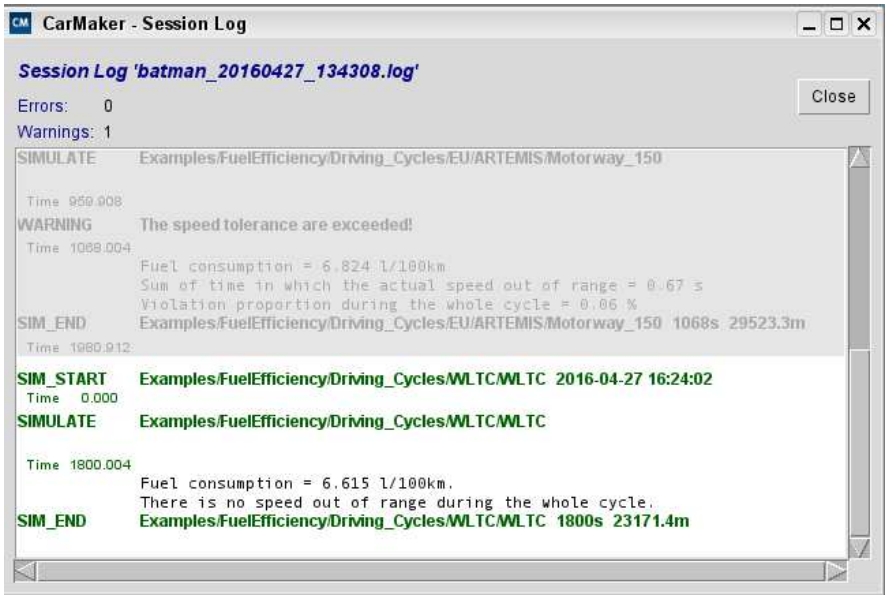


Figure 4.3: TestRun results of NEDC showed in Session Log

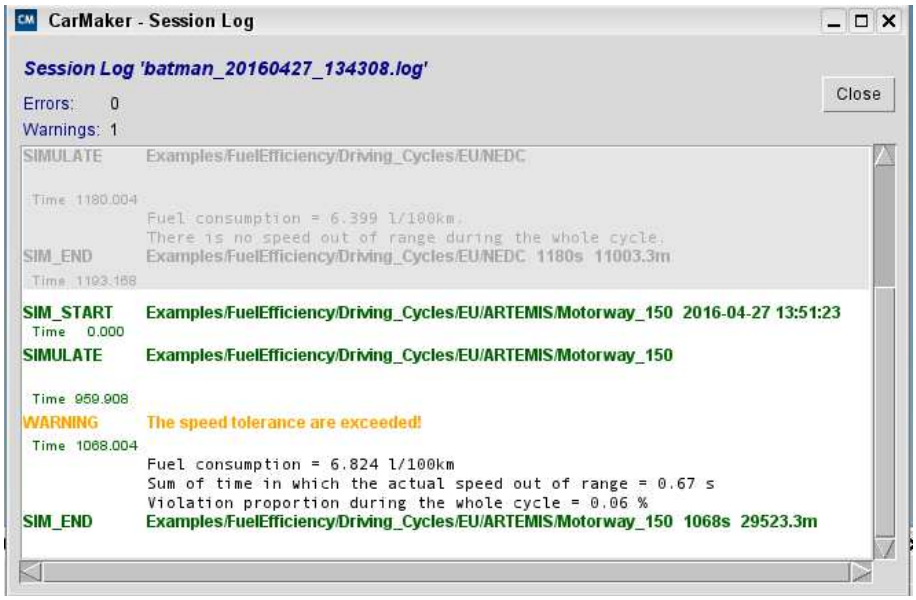


Figure 4.4: TestRun results of ARTEMIS Motorway 150 showed in Session Log

Attention! The vehicle parameters for the selected TestRun:Vehicle must be adhered to:

- Available fuel consumption map, which can be activated in *Vehicle Data Set > Powertrain > Drive Source > Engine > Fuel Consumption*
- The minimum engine speed for 1st gear to shift down must be greater than or equal to the engine idle speed.

Alternatively, the fuel consumption can also be seen in the IPGControl by selecting the appropriate quantities.

The conformation of the vehicle speed within the limits can be checked in IPGControl. A group called "Driving_Cycle" has been defined which contains the data such as the vehicle speed, the upper speed limit and the lower speed limit etc.. By right clicking on the data win-

dow and selecting Display Group > driving cycles, the user has information on the speed conformity. The results of a segment of the driving cycle WLTC class3b illustrated in IPG Control is shown in Figure 4.5.

The Test Manager can be expanded for future use by using the same pattern of implementation in case any other new driving cycle has to be added to the existing collection. In case of simple cycles, they can be implemented directly. In case of a more complicated approach for calculating the gear-shift strategy, simple Tcl/Tk programs can be written in the Script-Control command prompt and the necessary values can be calculated appropriately. For each of the new cycles, it has to be considered if the two different variations of manual and automatic transmission have to be added correspondingly, the data file for the IFF has to be changed so that it includes columns for both manual and automatic transmissions.

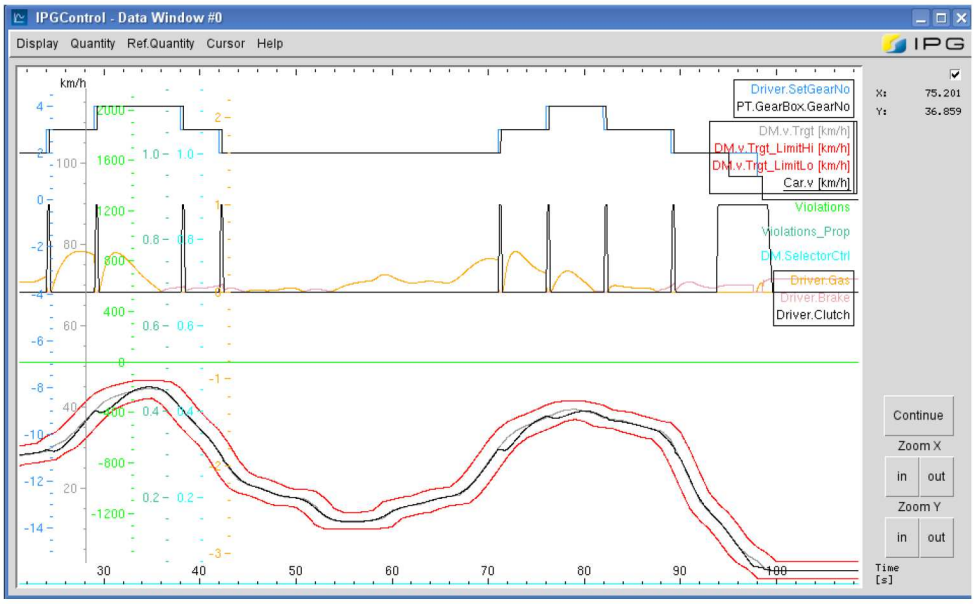


Figure 4.5: A segment of the TestRun results of driving cycle WLTC class3b in IPG Control

For demonstration purpose the DemoCar will normally be used to run the driving cycles with an exception to the driving cycle US06. In this case the vehicle Demo_AudiTT_Roadster will be used because of the "sport" namely aggressive characteristic of the cycles speed profile. According to Table 4.1 the maximum acceleration of the US06 driving cycle is up to 3.76 m/s^2.

For WLTC cycles:

For a plausible gear selection with regard to the WLTC cycles, we recommend a complete full load map in *MainGUI* > *Car* > *PowerTrain* > *DriveSource* > *Engine* > *Torque*. It means that the full load map should begin with idle speed where the engine torque is zero, and end by maximum speed with again zero torque. If this could not be done, please add following quantity under *Vehicle Data Set* > *Misc* > *Additional Parameter*:

```
PowerTrain.Engine.nMax = <max. engine speed>
```

The name "PowerTrain.Engine.nMax" should be written exactly the same.

In the following, important information (e.g. speed profiles, characteristics) about the different driving cycles is prepared for you.

4.1.1 Overview

There are a number of driving cycles that are being used around the world for the assessment of vehicle performance, fuel efficiency calculations and exhaust emission estimations. For the comparative testing of vehicle, the following driving cycles are implemented in CarMaker. The Drive Cycles are split in specific folders in the TestRun Examples, including cycles for EU, Japan, US and the WLTC. The table shows the driving cycle characteristics including e.g. distance, maximum speed and acceleration and further important informations, see Table 4.1. The comparison shows, e.g. that the US06 cycle has the highest demand of accelerating power (3.76 m/s^2).

Characteristics	Unit	NEDC	ARTEMIS				JC08
			Urban	Rural	Motorway 130	Motorway 150	
Distance	km	11.01	4.87	17.27	28.74	29.55	8.17
Total time	s	1180	993	1082	1068	1068	1204
Idle (standing) time	s	292	281	32	15	15	356
Average speed (incl. stops)	km/h	33.60	17.65	57.47	96.86	99.59	24.43
Average driving speed (excl. stops)	km/h	44.65	24.62	59.22	98.15	101.00	34.69
Maximum speed	km/h	120	57.70	111.50	131.80	150.40	81.60
Maximum acceleration	m/s^2	1.04	2.86	2.36	1.92	1.92	1.69
Characteristics	Unit	US-Cycles					WLTC (class 3b)
		FTP-72	FTP-75	US06	SC03	HWFET	NYCC	
Distance	km	11.99	17.77	12.89	5.76	16.51	1.90	23.27
Total time	s	1369	1874	600	600	765	598	1800
Idle (standing) time	s	257	356	43	115	4	208	233
Average speed (incl. stops)	km/h	31.53	34.14	77.33	34.56	77.68	11.42	46.53
Average driving speed (excl. stops)	km/h	38.82	42.14	83.30	42.76	78.09	17.51	53.45
Maximum speed	km/h	91.25	91.25	129.23	88.19	96.40	44.58	131.30
Maximum acceleration	m/s^2	1.48	1.48	3.76	2.28	1.44	2.68	1.67

Table 4.1: Driving Cycles for fuel consumption and emission tests

New European Driving Cycle (NEDC)

NEDC

The NEDC cycle is a synthetically developed cycle which is being used in the European Union since 1990 for testing and validation of exhaust emissions of new light duty vehicles. It consists of the ECE-15 (Elementary Urban Cycle) and the EUDC (Extra Urban Driving Cycle). The elementary urban cycle, each of duration 195 s, is repeated 4 times and the extra urban driving cycle lasts for 400 s, this leads to a entire duration of 1180 s (see Figure 4.6). A set of parameters, which represent the cycle features, is included in Figure 4.2.

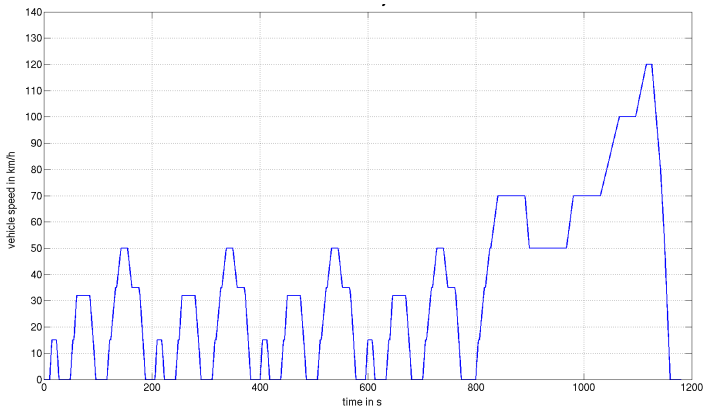


Figure 4.6: NEDC Cycle

Attention: By regulation, a speed trace tolerance of ±2 km/h combined with a time tolerance of ±1s is allowed when following the speed profile. For vehicles with automatic transmission, the speed trace must be followed as best as possible. For vehicles equipped with manual transmission, the regulation specifies the gear changes. Speed tolerances greater than those prescribed are accepted during phase changes, provided that the tolerances are never exceeded for more than 0,5 second on any one occasion.[1]

Characteristics	Unit	ECE 15	EUDC	NEDC
Distance	km	1.0146	6.9549	11.0132
Total time	s	195	400	1180
Idle (standing) time	s	62	40	292
Average speed (incl. stops)	km/h	18.73	62.59	33.60
Average driving speed (excl. stops)	km/h	27.46	69.55	44.65
Maximum speed	km/h	50	120	120
Maximum acceleration	m/s^2	1.042	0.833	1.042

Table 4.2: Summary of selected parameters for the NEDC driving cycle

ARTEMIS Driving Cycle

to be documented

Urban

to be documented

Rural

to be documented

Motorway_130

to be documented

Motorway_150

The ARTEMIS was commissioned under the European Commission 5th Framework project and during the project, a series of driving cycles were developed using representative real-world driving patterns, of which four cycles (Urban, Rural, Motorway 130 and Motorway 150), are the most commonly used cycles. The v-t curve of the four cycles are accordingly illustrated from Figure 4.7 to Figure 4.10. A summary of the selected parameters for these four cycles are shown in Table 4.3.

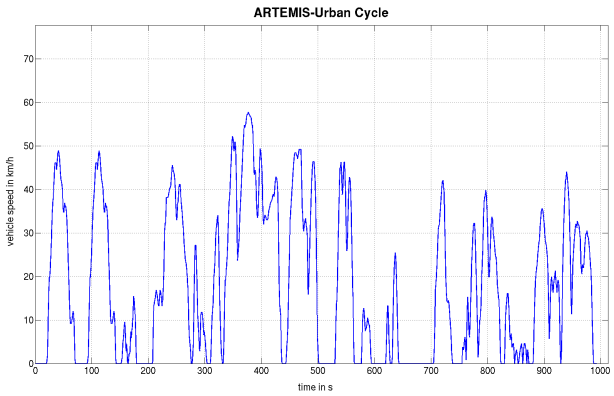


Figure 4.7: Speed profile (v-t) of ARTEMIS-Urban Driving Cycle

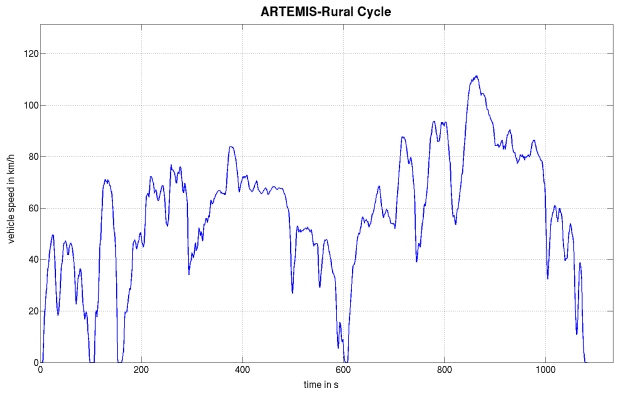


Figure 4.8: Speed profile (v-t) of ARTEMIS-Rural Driving Cycle

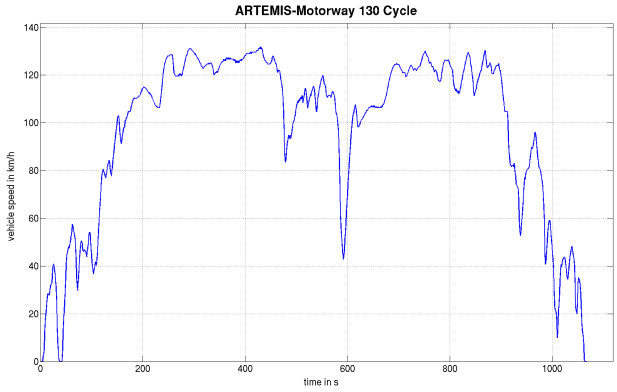


Figure 4.9: Speed profile (v-t) of ARTEMIS-Motorway 130 Driving Cycle

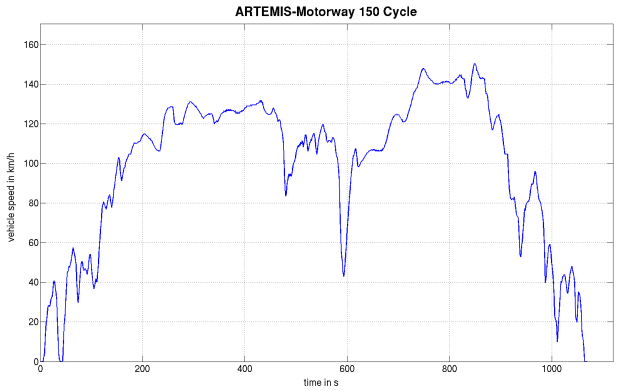


Figure 4.10: Speed profile (v-t) of ARTEMIS-Motorway 150 Driving Cycle

Attention: For all the ARTEMIS Cycles, the official regulations allow the driver a speed tolerance of ±2 km/h combined with time tolerance of ±1s when following the speed profile. A test should be accepted if it is within these limits for greater than 99% of the time, and if the driven distance is within 1% of the reference distance. [2]

Characteristics	Unit	Urban	Rural	Motorw. 130	Motorw. 150
Distance	km	4.8698	17.2725	28.7360	29.5450
Total time	s	993	1082	1068	1068
Idle (standing) time	s	281	32	15	15
Average speed (incl. stops)	km/h	17.65	57.47	96.86	99.59
Average driving speed (excl. stops)	km/h	24.62	59.22	98.15	101.00
Maximum speed	km/h	57.70	111.50	131.80	150.40
Maximum acceleration	m/s^2	2.861	2.361	1.917	1.917

Table 4.3: Summary of selected parameters for the ARTEMIS cycles

In addition, the regulations provide the specific gear changes that need to be made for vehicles equipped with manual transmission during the course of the driving schedule. In this use case with CarMaker, the "Cycle" Strategies [3] are implemented. It suggests four different types of gear-change strategies. The power-to-weight ratio along with the top-speed of the vehicle in third gear decide under which class a particular vehicle falls (The classification and the needed calculations please see ARTEMIS Driving Cycle).

JC08

Japanese Driving Cycle (JC08)

This is the cycle prescribed by the Ministry of Land, Infrastructure, Transport and Tourism Japan for the testing of vehicles whose unloaded weight is less than 3500kg (Lightweight vehicles). The test represents driving in congested city traffic, including idling periods and frequently alternating acceleration and deceleration. The test is used for emission measurement and fuel economy determination, for gasoline and diesel vehicles. The speed profile is as shown in the Figure 4.11 below. In addition, a summary of selected parameters for this cycle is included in Table 4.4.

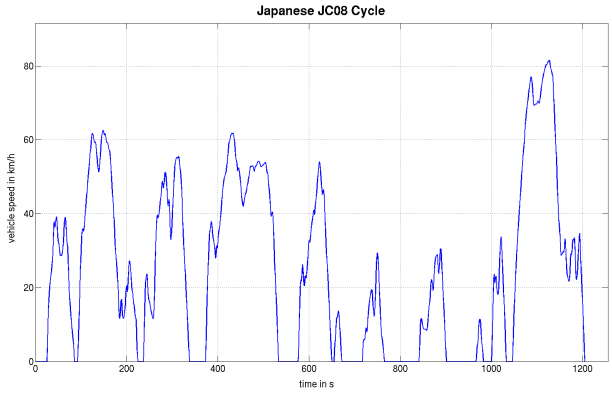


Figure 4.11: Speed profile (v-t) of JC08 Driving Cycle

Characteristics	Unit	JC08
Distance	km	8.1721
Total time	s	1204
Idle (standing) time	s	356
Average speed (incl. stops)	km/h	24.43
Average driving speed (excl. stops)	km/h	34.69
Maximum speed	km/h	81.60
Maximum acceleration	m/s^2	1.69

Table 4.4: Summary of selected parameters for the JC08 cycle

Another cycle called the JE05 [5] cycle is also prescribed by the Ministry for the testing of vehicles whose unloaded weight is greater than 3500kg (heavy vehicles). This cycle is not implemented in the current test manager.

Note to the speed tolerance: A speed trace tolerance of 2 km/h combined with a time tolerance of 1s is allowed when following the speed profile as per the regulations. Speed tolerances greater than prescribed shall be accepted provided the tolerances are not exceeded for more than 2 seconds during the whole cycle [5].

US Driving Cycles

There are varieties of driving cycles used by the US government. Some of them are used for general fuel consumption and emission test, while some others are specific driving cycles exclusively for congested city or fluent highway traffic. Generally, only the v-t profile are explicitly given and no gear strategies in detail are given.

FTP-72

The FTP 72 (Federal Test Procedure) was the first driving cycle that was used by the US government to validate the exhaust emission of new light duty vehicles. The cycle simulates an urban route and it consists of two phases. Phase 1 begins with a cold start phase (duration 505s) and phase 2 is transient (duration 864s). The speed profile is as shown below in Figure 4.12.

FTP-75

The FTP 75 cycle is an extension of the FTP 72 cycle. It is the current official driving cycle that is being used by the US Government. The cycle consists of the same two phases as the FTP 72 cycle and there is a phase three, which is a repetition of phase one for the same duration but in a hot start condition. The speed profile is as shown below in Figure 4.13.

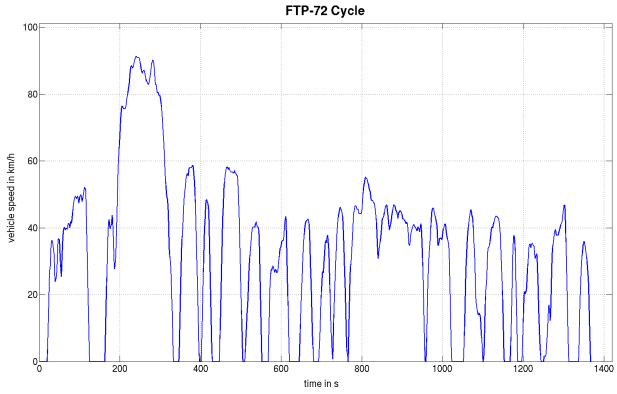


Figure 4.12: Speed profile (v-t) of the FTP-72 Driving Cycle

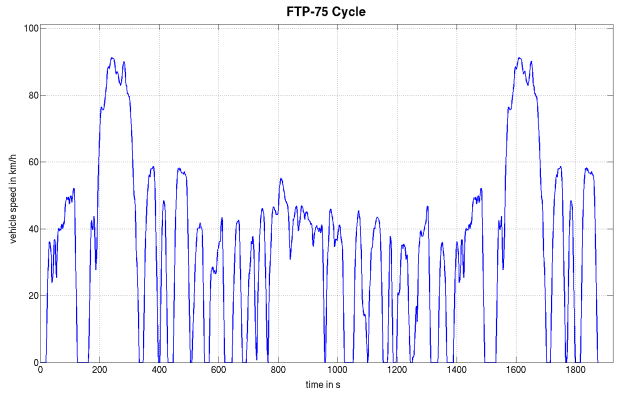


Figure 4.13: Speed profile (v-t) of the FTP-75 Driving Cycle

However, to address the shortcomings of the FTP 75 cycle, other cycles are used along with the FTP 75 cycle. These are as follows:

US06

The US-06 Driving Cycle simulates an aggressive driving behavior with sharp accelerations and decelerations and rapid speed fluctuations. These characteristics are well illustrated in Figure 4.14. It is worth noting that the maximum acceleration during the cycle is about 3.76 m/s^2.

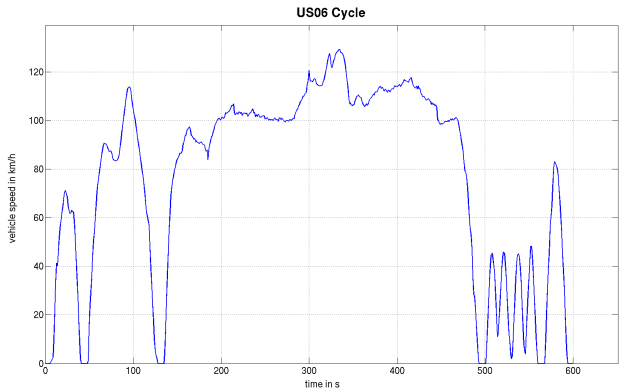


Figure 4.14: Speed profile (v-t) of the US06 Driving Cycle

SC03

The SC03 Supplemental Federal Test Procedure (SFTP) has been introduced to represent the engine load and emissions associated with the use of air conditioning (A/C) in vehicles certified over the FTP 75 test cycle. The SC03 is a chassis dynamometer test performed with the vehicle A/C unit turned on, at a lab temperature of 35˚C. The cycle has an average speed of 34.8 km/h and duration of 596 seconds.

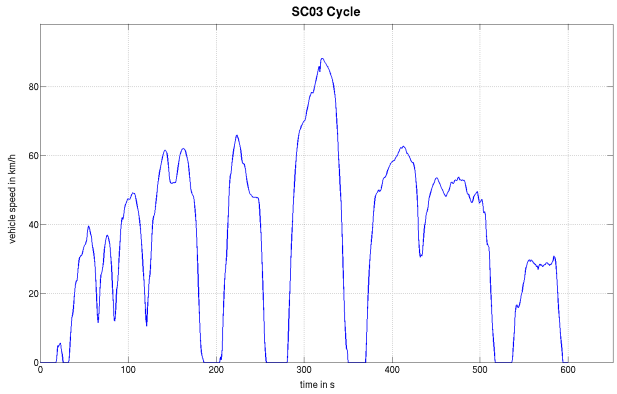


Figure 4.15: Speed profile (v-t) of the SC03 Driving Cycle

HWFET

The HWFET is used to determine the highway fuel economy rating, while the city rating is based on the FTP-75 test.

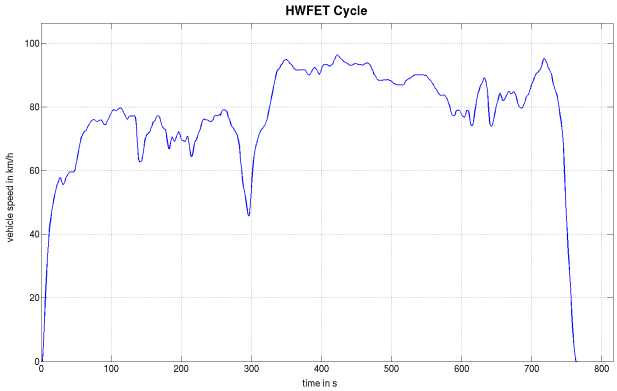


Figure 4.16: Speed profile (v-t) of the HWFET Driving Cycle

NYCC

The test simulates low speed urban driving with frequent stops within the New York City.

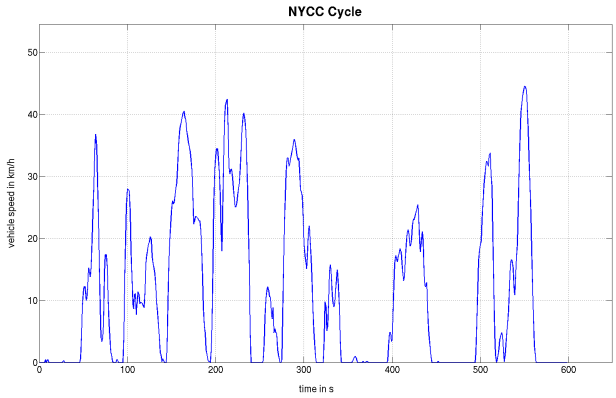


Figure 4.17: Speed profile (v-t) of the NYCC Driving Cycle

InTable 4.5 some typical parameters to characterize the US driving cycles are summarized.

Characteristics	Unit	FTP-72	FTP-75	US06	SC03	HWFET	NYCC
Distance	km	11.9902	17.7694	12.8876	5.7606	16.5065	1.8973
Total time	s	1369	1874	600	600	765	598
Idle (standing) time	s	257	356	43	115	4	208
Average speed (incl. stops)	km/h	31.53	34.14	77.33	34.56	77.68	11.42
Average driving speed (excl. stops)	km/h	38.82	42.14	83.30	42.76	78.09	17.51
Maximum speed	km/h	91.25	91.25	129.23	88.19	96.40	44.58
Maximum acceleration	m/s^2	1.48	1.48	3.76	2.28	1.44	2.68

Table 4.5: Summary of selected parameters for the US driving cycles

Note to the speed tolerance:

(1) The upper limit is 2 mph (3.2 km/h) higher than the highest point on the trace within 1 second of the given time.

(2) The lower limit is 2 mph (3.2 km/h) lower than the lowest point on the trace within 1 second of the given time.

(3) Speed variations greater than the tolerances (such as may occur during gear changes or braking spikes) are acceptable, provided they occur for less than 2 seconds on any occasion and are clearly documented as to the time and speed at that point of the driving schedule.[6]

Global Cycle: World harmonized Light vehicle Test Cycle

WLTC

The WLTP cycle is being developed by the EU, Japan, and India under the guidance of the UNECE World Forum for Harmonization of Vehicle Regulations in order to harmonize the vehicle emission test procedures and performance requirements of light duty vehicles as much as possible on the global scale. The following cycles are intended to represent a more real-world simulation of driving compared to the NEDC cycle. The developed cycles are categorized based on the power to weight ratio of the car. They are as follows:

Category	PMR (Power-Mass-Ratio)*	Speed Phases
Class 3b	PMR > 34 & v_{max}* >120	Low3, Medium 3-2, High3-2, Extra High3
Class 3a	PMR > 34 & v_{max} <120	Low3, Medium 3-1, High3-1, Extra High3
Class 2	34 ≥ PMR > 22	Low2, Medium 2, High2, Extra High2
Class 1	PMR ≤ 22	Low1, Medium1

Table 4.6: The Category of WLTC Cycles in dependence of PMR and maximum vehicle speed (PMR: Power to mass ratio)

Depending on the maximum speed that can be attained by the vehicle, there are different combinations of the low, medium, high and extra high-speed phases. The speed profile in different cycle phases varies for different classes. For example, the low speed phase in class 1 is different from the low speed phase in class 3. However, the duration of the low speed phase in both the mentioned classes remains the same (i.e. 589 s). Hence, the characteristics such as total distance travelled, average speed etc. also depend on the classes and the phases that the cycle consists of. That makes it in total four different cycles under the WLTC group. In fact, most of the individual vehicles falls into the group class 3b. Some selected parameters for this class, for each phase and for the whole cycle as well, are summarized in Table 4.7 as below. In addition, the v-t profile of it, split into the four phases, are illustrated from Figure 4.18 to Figure 4.21.

Characteristics	Unit	Low	Medium	High	Extra High	Total
Distance	km	3.0945	4.7559	7.1617	8.2541	23.2663
Total time	s	589	433	455	323	1800
Idle (standing) time	s	148	47	29	6	233
Average speed (incl. stops)	km/h	18.91	39.54	56.66	92.00	46.53
Average driving speed (excl. stops)	km/h	25.26	44.36	60.52	93.74	53.45
Maximum speed	km/h	56.50	76.60	97.40	131.30	131.30
Maximum acceleration	m/s^2	1.61	1.61	1.67	1.61	1.67

Table 4.7: Summary of selected parameters for the WLTC cycle class 3b

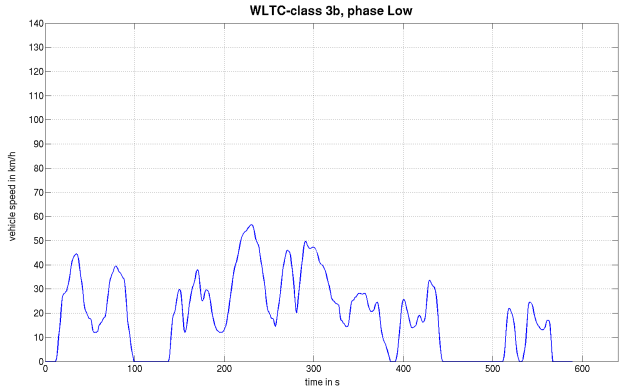


Figure 4.18: Speed profile (v-t) of the WLTC Cycle class 3b phase low

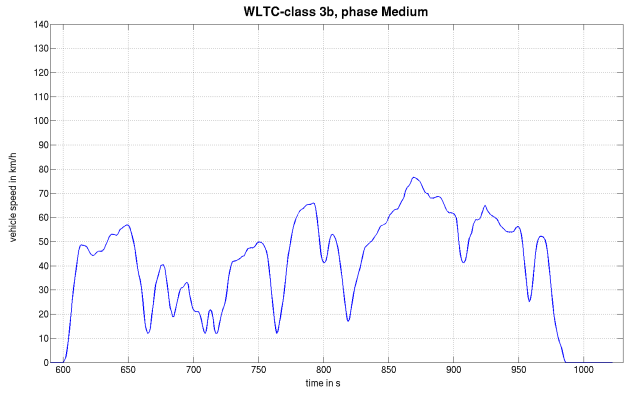


Figure 4.19: Speed profile (v-t) of the WLTC Cycle class 3b phase medium

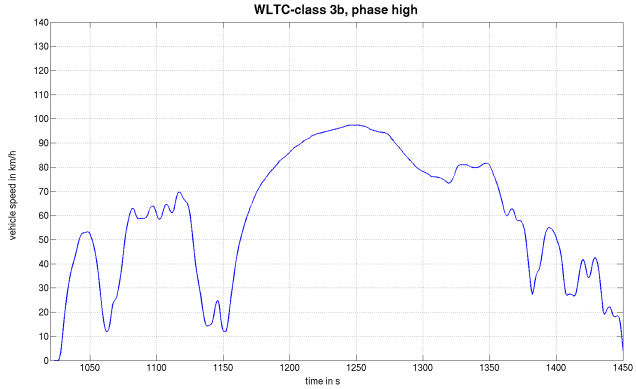


Figure 4.20: Speed profile (v-t) of the WLTC Cycle class 3b phase high

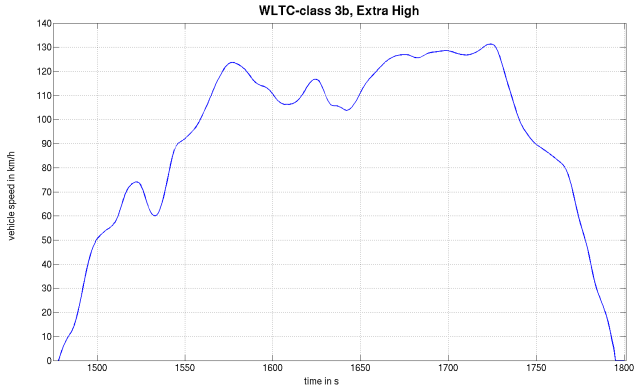


Figure 4.21: Speed profile (v-t) of the WLTC Cycle class 3b phase extra high

The gear strategy for vehicle depends on various parameters of the vehicle and they are calculated based upon the formula that are given in the regulations [8]. Details to the gear strategy is explained in the section Details on Gear Selection process for WLTC.

Note to the speed tolerance: The tolerances on speeds are 2 km/h in combination with a time tolerance of one second. Speed tolerances greater than prescribed shall be accepted provided the tolerances are never exceeded for more than one second on any occasion and

no more than ten such deviations per test. In addition, considering the driveability, down-scaling of the speed especially in high or extra high phase of all classes can be performed when needed.

4.1.2 Details on Implementation

The driving cycles which are mentioned in the last chapter are implemented with CarMaker Test Manager. Details will be explained in the following section.

Global Settings:

The control over the TestRuns in Test Manager is realized within a tcl script which includes the relevant functions for: start process, the end process and the gear selection procedure for driving cycles like ARTEMIS and WLTC. The script file "**DrivingCycles.tcl**" is saved in the folder *<Installation Directory>/Examples/Powertrain/DrivingCycles/Scripts*. An overview of the structure in the tcl file is illustrated in Figure 4.22.

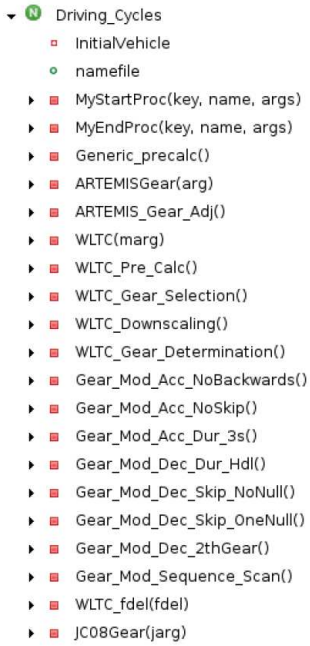


Figure 4.22: Structure of the main tcl file for the DriveCycle implementation in TestManager

The *DemoCar* is selected as the default car for the TestRuns. The car that needs to be tested is set in the initial settings just once and then all the cycles are adapted to the car that is selected. The car is selected by varying the KValue "**TestRun:Vehicle**" and giving the path of the required vehicle info-file. Besides that user-defined Key Values (KValues) or Named Values (NValues) can be specified on the parameters tab for changing certain properties of the chosen vehicle, the driver or some other relevant parameters. Some predefined parameters are listed in the TestManager as below in Figure 4.23.

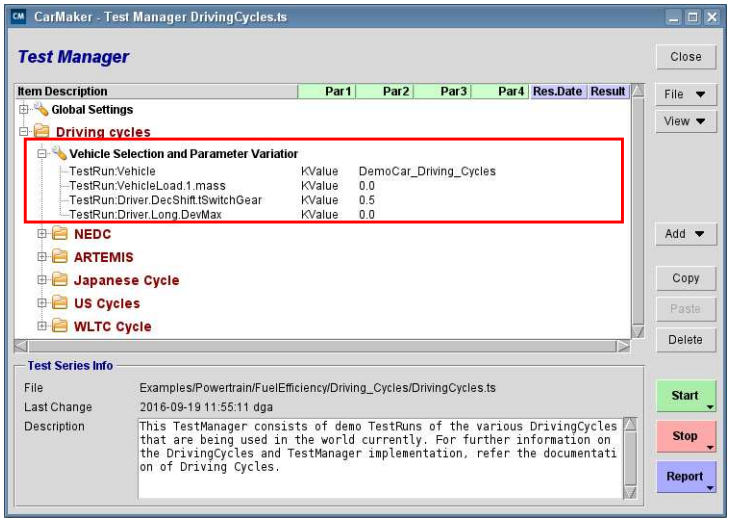


Figure 4.23: Vehicle parameters and Parameter Variator

TestRun:Vehicle: The user-defined vehicle for the driving cycle can be set here to replace the original vehicle defined in the TestRun.

In case of a user-defined vehicle, it must be ensured that the minimum engine speed for the first gear to shift down (Declutching/Gear Shifting dialog box of IPGDriver) must be greater than or equal to the engine idle speed. Otherwise it occurs a system error.

TestRun:VehicleLoad.1.mass: Masses to setup TestRun mass contribution of the vehicle.

TestRun:Driver.DecShift.tSwitchGear: Time for Shifting of IPGDriver, which affects the speed following behavior significantly.

TestRun:Driver.Long.DevMax: This parameter describes the threshold that IPGDriver uses to make corrections with the gas and brake pedals. If the absolute value of the predicted speed deviation is lower than this threshold, IPGDriver activities are kept to a minimum. For Driving Cycles, which require the given speed profile to be kept as exact as possible the threshold value should be set to 0.0.

The user can naturally add or remove the parameters defined here to influence the behavior of all driving cycles. Otherwise, changes to a certain parameter for a specific driving cycle can be done by directly adding a variation under the corresponding TestRun.

TestRun Groups

For each Driving Cycle, a TestRun is set up. In the TestManager, it is possible to categorize the cycles into their respective groups. Here, each of the cycles is implemented under their main heading such as NEDC, WLTC, US Cycles and Japanese cycles. Under these headings, the different types of the cycles are added. For example, the 6 different driving cycles within the US Cycles are grouped under a single heading titled "US" and variations can be added to vary the TestRun.

In the TestRuns, the longitudinal dynamics is controlled by the Speed Profile with the function Input From File (IFF), see Figure 4.31. The data file used for the IFF contains the time-speed data points, the upper and lower velocity limits and separate columns either for automatic transmission gear changes or for manual transmission gear selection. This decision is made using ScriptControl programming. For the selection of the appropriate gear-change column from the data file, the KeyValue of the parameter "DrivMan.OW.Quantities" will be manipulated. Details to this topic will be explained in the following chapters for each specific driving cycle group.

Because the procs in TM_Driving_Cycles.tcl are defined under the namespace Driving_Cycles, therefore, when you want to select a certain proc for a certain TestRun, the ScriptControl commands in Test Manager should begin with Driving_Cycles::<proc>, e.g.: Driving_Cycles::WLTC.

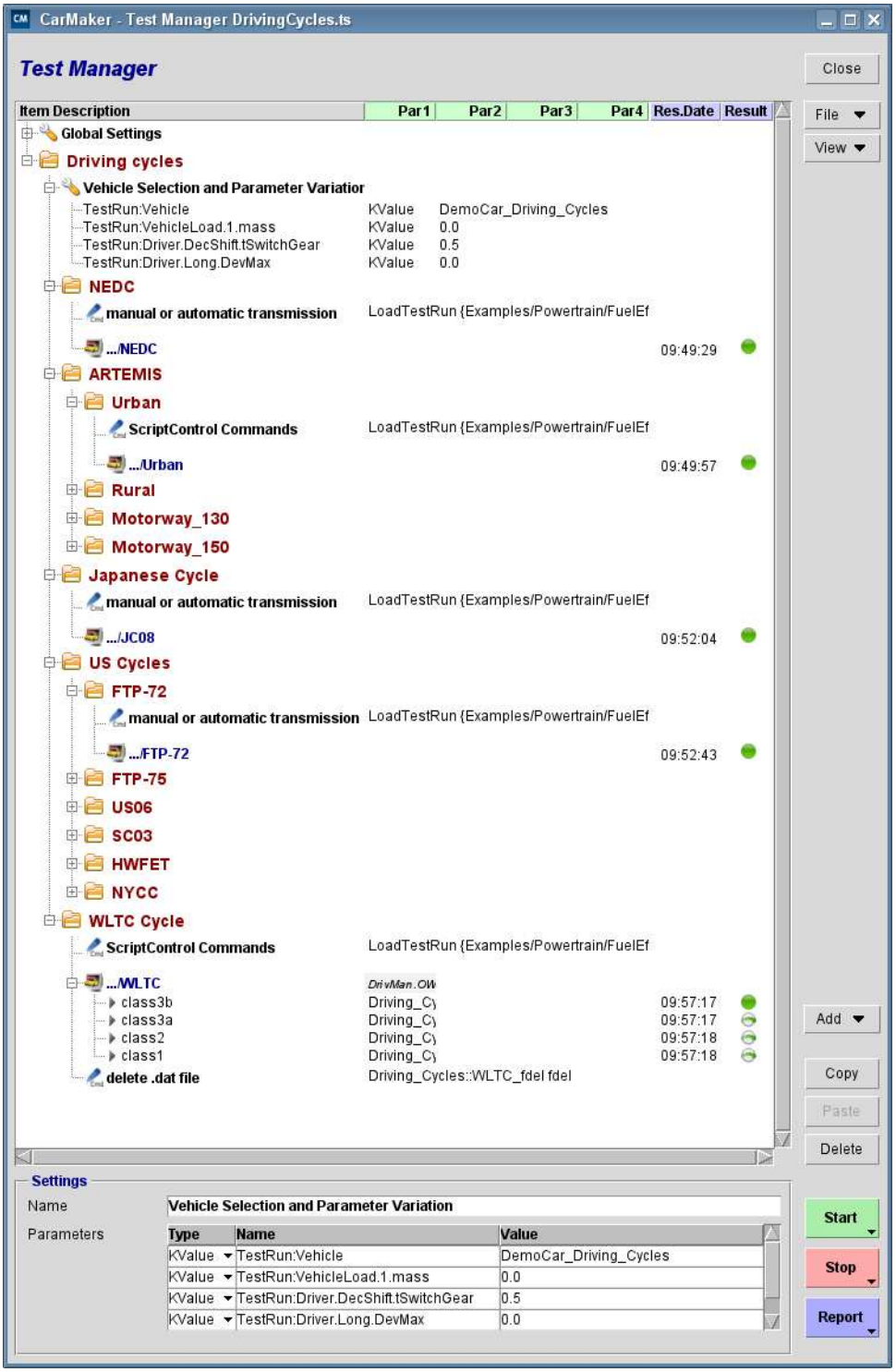


Figure 4.24: Overview of the Implementation of the DrivingCycles in TestManager

NEDC

From regulation to .dat file for IFF

There is no explicit time-velocity data points for each second in NEDC technical regulation. However, as we can see from the Figure 4.6 that the velocity of the cycle changes straight proportional.

At first, we discuss the elementary urban cycle. According to Table 4.10, we can obtain the time-velocity data points for each second by means of interpolation. For example, in the second phase, (see Table 4.10) the acceleration from 0-15 km/h is operated, and the duration is 4s. Therefore, the time-velocity data points from 11s to 15s are:

Time (s)	Velocity (km/h)
11	0
12	3.75
13	7.5
14	11.25
15	15

Table 4.8: Interpolation result for acceleration from 11s to 15s global time of the NEDC cycle

As to the case of a manual gearbox, the gear to be used is also given in the regulation. There are certain gear change operations. For example, the 8th operation with a duration of 2s is for gear change from 1st gear to 2nd gear. In practice, the IPGDriver should be told that after the 7th phase a gear change should be operated. Considering the "Time for shifting" in IPGDriver, which in this case is set to 1s, the last second of the operation before should also set to the gear to be changed, in order to catch up with the cycle. E.g., the gears to be used from 54s to 57s (from 7th operation to 9th operation) are:

Operation	Time (s)	Gear to be used
7 Acceleration	53	1
7 Acceleration	54	2
8 Gear Change	55	2
8 Gear Change	56	2
9 Acceleration	57	2

Table 4.9: Result for the Gearselection from 53s to 57s of the NEDC cycle

Note: In the deceleration phase to stand still, the operation of clutch is also defined in the regulation. E.g., in the 5th phase the last 3s of the deceleration to 0 km/h the clutch is disengaged. However, in the implementation this task is left to the IPGDriver.

No of operation	Operation	Phase	Acceleration (m/s^2)	Speed (km/h)	Duration of each		Cumulative time (s)	Gear to be used in the case of a manual gearbox
					Operation (s)	Phase (s)		
1	Idling	1			11	11	11	6 s PM + 5 s K_1 (*)
2	Acceleration	2	1,04	0 – 15	4	4	15	1
3	Steady speed	3		15	9	8	23	1
4	Deceleration		−0,69	15 – 10	2		25	1
5	Deceleration, clutch disengaged	4	−0,92	10 – 0	3	5	28	K_1 (*)
6	Idling	5			21	21	49	16 s PM + 5 s K_1 (*)
7	Acceleration		0,83	0 – 15	5		54	1
8	Gear change	6			2	12	56	
9	Acceleration		0,94	15 – 32	5		61	2
10	Steady speed	7		32	24	24	85	2
11	Deceleration		−0,75	32 – 10	8		93	2
12	Deceleration, clutch disengaged	8	−0,92	10 – 0	3	11	96	K_2 (*)
13	Idling	9			21	21	117	16 s PM + 5 s K_1 (*)
14	Acceleration		0 – 15	0 – 15	5		122	1
15	Gear change				2		124	
16	Acceleration	10	0,62	15 – 35	9	26	133	2
17	Gear change				2		135	
18	Acceleration		0,52	35 – 50	8		143	3
19	Steady speed	11		50	12	12	155	3
20	Deceleration	12	−0,52	50 – 35	8	8	163	3
21	Steady speed	13		35	13	13	176	3
22	Gear change				2		178	
23	Deceleration	14	−0,86	32 – 10	7	12	185	2
24	Deceleration, clutch disengaged		−0,92	10 – 0	3		188	K_2 (*)
25	Idling	15			7	7	195	7 s PM (*)

(*) PM = gearbox in neutral, clutch engaged.
K_1, K_2 = first or second gear engaged, clutch disengaged.

Table 4.10: Operation of Elementary Urban Cycle [1]

The same treatment will be applied to the "Extra Urban Cycle". According to the regulation, for vehicles with a maximum engine power/mass ratio in running order of less than or equal to 40 kW per tonne, and a maximum speed less than or equal to 130 km/h, the maximum speed of the extra-urban cycle shall be limited to 90 km/h. The extra-urban cycle will be modified to the one as shown in Figure 4.25. This case with underpowered vehicle is currently not considered.

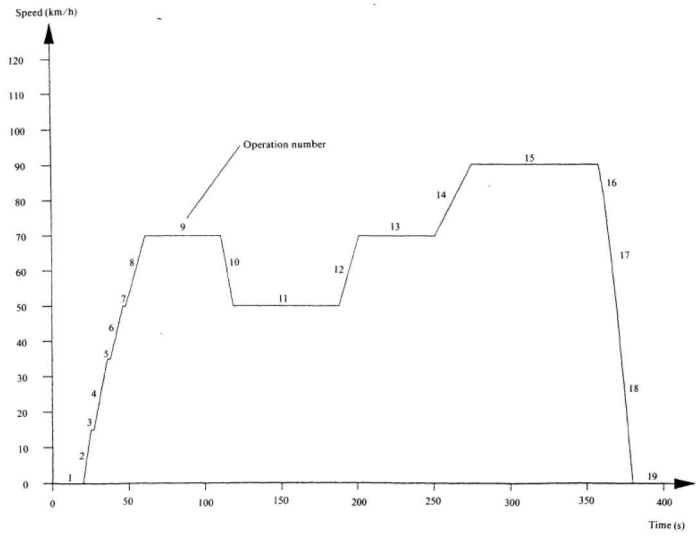


Figure 4.25: Extra-urban cycle for underpowered vehicles [1]

Note: In general, in case of downward gear change from the gear higher than 1 directly to neutral, the actual gear in CM powertrain cannot strictly follow the given set from IPGDriver. However, during downshift the clutch is steadily disengaged. This means the gear change has no impact on the longitudinal dynamic of the vehicle. Accordingly, the downshift operation is equatable to a direct shift from higher gear to neutral.

Implementation in Test Manager

For the NEDC cycle, a single TestRun is added. The transmission type of the vehicle that is defined in test series, will be read by DrivingCycles.tcl and either "GearNo" column or the "SelectorCtrl" column in IFF (Input from File) will be automatically selected:

- If it is an automatic transmission, the "SelectorCtrl" column from the data file of IFF will be selected
- If it is a manual transmission, the "GearNo" column from the data file of IFF will be selected

Note: The regulation only suggest a gear shifting strategy up to the 5th gear. It might not be appropriate to use this strategy for a vehicle with for example 7 gears. It is recommended by the regulation that a modified gear strategy should be developed together with manufacturer. Therefore, for this Test Manager Implementation, we let the IPGDriver make the gear shifting decision freely, when the vehicle is equipped with more than 5 gears.

ARTEMIS Driving Cycle

From regulation to .dat file for IFF

General Information of the driving cycle can be found in [2], [3]. When it comes to a vehicle with a manual transmission, the "Cycle" gear strategies should be included in the .dat file. IPG got the explicit time-velocity data points and the gear strategies directly from IFSTTAR - LTE Laboratory Transports & Environment in France.

Gear strategy selection

The classification and the needed calculations for the "Cycle" gear strategies for a specific vehicle are defined in [3]. It will be shortly introduced below.

At first the variables needed for the classification are:

P: the maximal power of the car (kW);

M: mass of the empty car (kg);

V(3)1000: vehicle speed at 1000 rpm in the 3rd ratio (km/h);

rpm: engine speed at the maximal power;

With these variables the 2 following parameters are calculated (with the right units):

MP = P/M, the vehicle massic power, in (W/kg)

V(3)P = V(3)1000 * rpm / 1000 the vehicle speed at the maximum power, in 3rd ratio, in (km/h)

With MP and V(3)P the following classifications are defined for the "Cycle" strategies:

If:	And:	Then choose strategy:
MP > 76 W/kg	*V(3)P > 110 km/h*	1
MP < 76	*V(3)P > 118*	2
MP < 60	*V(3)P < 102*	4
In all other cases:		3

Table 4.11: The Gear Change Strategies Category of ARTEMIS Cycles [3]

When a particular vehicle with manual transmission in Test Manager is selected, the calculation of MP and V(3)P will be done. With these two criteria, the appropriate gear strategy for this vehicle can then be selected. This procedure is realized through the proc ARTEMIS-Gear in TM_Driving_Cycles.tcl. The corresponding line in the ScriptControl Command "manual or automatic transmission" in TestManager is (2)

```
1:  # script control command for gear selection of ARTEMIS cycle
```

```
2:  Driving_Cycles::ARTEMISGear
```

Modification of gear strategies

When we run the Urban Cycle using the given gear strategy with DemoCar (in this case the gear strategy 1 will be executed), we get the results (see Figure 4.26).

The reasons for this high "out of range" proportion are mainly the following:

1. Acceleration from standstill: The change from neutral to 1st gear is too late. One second earlier is better.

2. In case of a very short deceleration to standstill (e.g. 3s), gears are original set to 0 (with clutch engaged). In CarMaker the equivalent version, namely the gears are set to 1st gear with clutch disengaged, will be implemented (e.g. the gear sequence 1-1-0-0-0-1-1 will then be 1-1-1-1-1-1-1).

When we run the Rural Cycle, we get similar results. As we look into the v-t and gear-clutch curves, we notice the same problem as described above. Since this is a general issue when it turns to the application of the "Cycle" strategies in CarMaker, an adjustment of the strategies will be made to achieve a better behavior. The corresponding code is written in proc ARTEMIS_Gear_Adj. With this adjustment, we get the following results as seen in Figure 4.27. With some more slightly changes to the gear strategy, according to the resulting curves, we can obtain a final gear strategy for a particular vehicle.

In proc ARTEMIS_Gear_Adj the original .dat file with cycle strategies, which are saved in the "ARTEMIS_Std" folder, will be read and modified. Therefore, DO NOT DELETE the "ARTEMIS_Std" folder in any case.

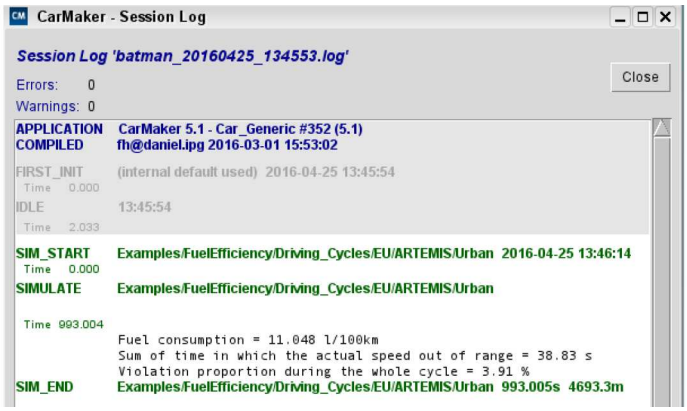


Figure 4.26: ARTEMIS Urban Driving Cycle with original cycle strategy

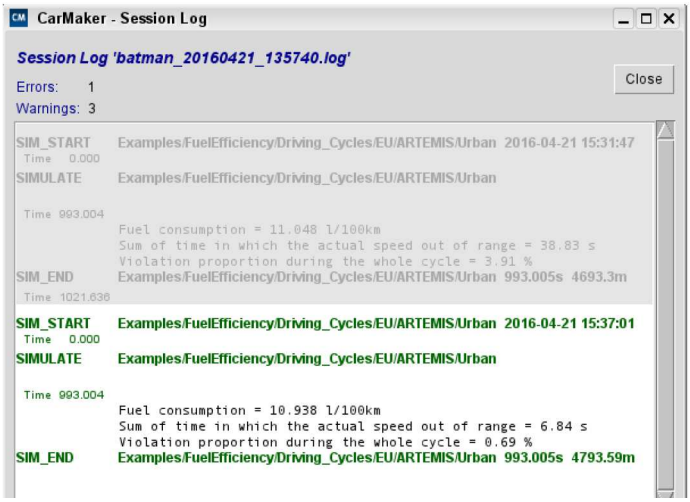


Figure 4.27: ARTEMIS Urban Driving Cycle with adjusted cycle strategy

Implementation in Test Manager

For the ARTEMIS cycle, a single TestRun is added. The transmission type of the vehicle that is defined in test series, will be read by DrivingCycles.tcl and either "GearNo" column or the "SelectorCtrl" column in IFF will be automatically selected:

- If it is an automatic transmission, the "SelectorCtrl" column from the data file of IFF will be selected
- If it is a manual transmission, the "GearNo" column from the data file of IFF will be selected

Japanese Cycle

From regulation to .dat file for IFF

The explicit time-velocity data points together with the gear strategies can be derived from [4]. The Japanese cycle JC08 prescribes two different gear strategies based on the unloaded weight and number of persons for those individual vehicles, which are not equipped with overdrive. The latter case is not considered in this implementation.

Implementation in Test Manager

For the JC08 cycle, a single TestRun is added. The gear strategy for particular vehicle is appropriately chosen by running the script file DrivingCycles.tcl just before simulation starts. The transmission type of the vehicle that is defined in test series, will also be read by DrivingCycles.tcl and either "GearNo" column or the "SelectorCtrl" column in IFF will be automatically selected:

- If it is an automatic transmission, the "SelectorCtrl" column from the data file of IFF will be selected
- If it is a manual transmission, the "GearNo" column from the data file of IFF will be selected

US Driving Cycles

From regulation to .dat file for IFF

The explicit time-velocity data points can be read out in [6] or directly downloaded from the official website of EPA (US Environmental Protection Agency) [7].

Implementation in Test Manager

For the US cycle, only the "SelectorCtrl" column is present in the data file. There is no regulation for a particular shift strategy for manual transmissions. So the vehicle needs to be driven as best as possible to follow the speed profile. Hence, the control of gears for the manual transmission is left to the IPGDriver.

WLTC

From regulation to .dat file for IFF

The WLTC cycle has four different types of cycles as described in Figure 4.6. The category, under which a particular vehicle falls into, is decided based upon the top speed of the vehicle and the power-to-weight ratio of the car. The explicit time-velocity data points for each type of cycle (without Downscaling) are given in [8]. Figure 4.28 shows a segment of the Implementation of time-velocity data points of class 3b in WLTC_class3b.dat file for IFF.

In case of downscaling, special treatment will be done in certain time period. This will be explained in Details on Gear Selection process for WLTC of this document.

For those vehicles equipped with manual transmissions, special regulations are developed by the WLTP working Group. Based on the informal document (version 2016) of the working group [8], the gear selection and shift point determination for these vehicles is implemented using tcl programming. A detailed description of the implementation is given in the section Details on Gear Selection process for WLTC of this document.

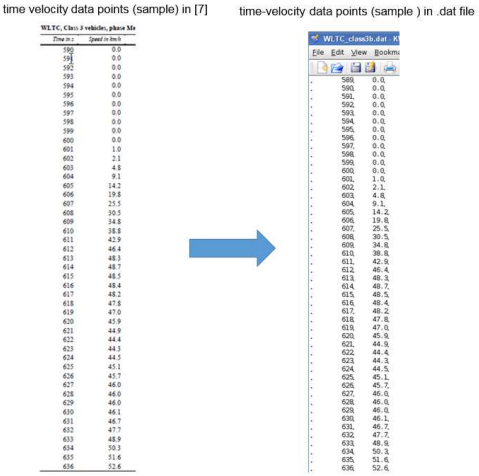


Figure 4.28: Implementation of v-t data points of class 3b in WLTC_class3b.dat file for IFF

Implementation in TestManager

With the tcl programming an appropriate gear strategy will be generated. The generated gear strategy is written into .dat file, which is later used as the data file for IFF by using the "KeyValue set" command to modify the parameter "DrivMan.OW.FName" (see Figure 4.29). For each time when the vehicle is changed, the relevant calculations are executed and only the data file for one of the four class-variations (3b, 3a 2, 1), under which the vehicle falls, is generated. Then, only the relevant driving cycle is simulated. The generated .dat file will later be deleted after the simulation has completed.

The name of TestRun (e.g. class3b) and the Value of "DrivMan.OW.FName" under the "TestRun Description" CAN NOT be modified. Otherwise, the data file for IFF cannot be read out correctly, an error massage will pop out to tell you that the data File cannot be opened.

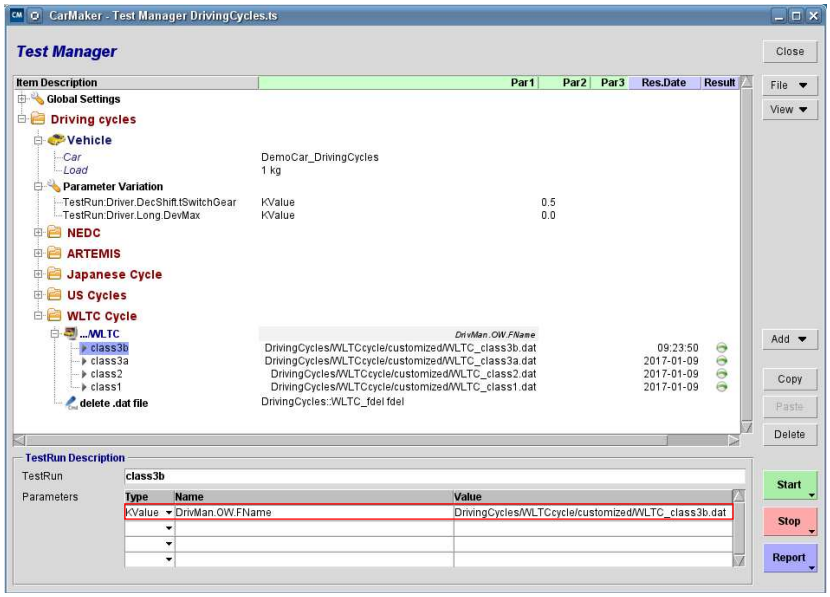


Figure 4.29: Special process in the WLTC Driving Cycles

CarMaker Functions

Fuel Consumption Map

One of the most important functions is given by the engine model. It contains a fuel consumption map which can be adjusted by the table on the left side (see Figure 4.30). The input for the diagram is rotational speed, torque and fuel consumption (specific [g/kWh] or absolute [g/s]). CarMaker uses the heating value (calorific value) of the fuel for calculating the current fuel consumption as well as the average consumption.

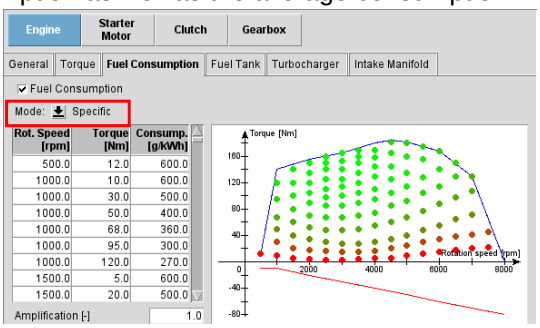


Figure 4.30: Fuel consumption map

The input consists of a set of speed points over the time. Along with lower and upper speed boarders and the gear number, see Figure 4.31. First line of inputfile shows

Time | Vel | VelUp | VelLow | GearNum

The IPGDriver follows the prescribed speed profile. (For more information see User's Guide chapter 'Input From File')

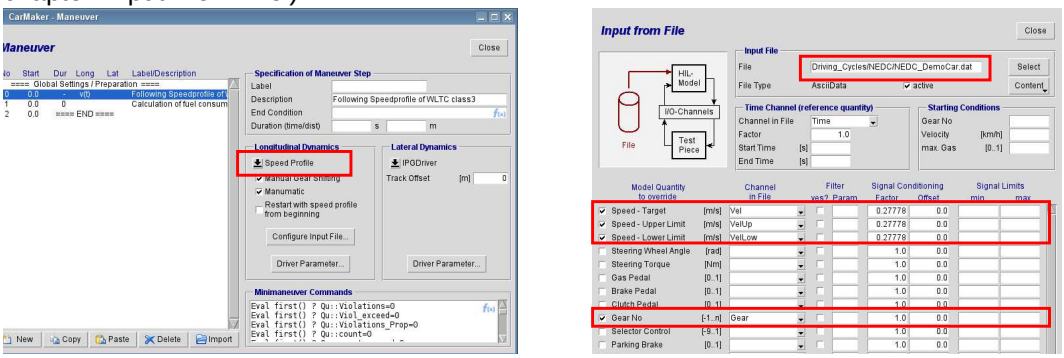


Figure 4.31: Vehicle control by file, left: maneuver selection; right: File and Input selection

4.1.3 Details on Gear Selection process for WLTC

The gear selection procedure is programmed with the tcl language. The source code can be found in Installation Directory > /Data/TestRun/Examples/Powertrain/DrivingCycles/Scripts/DrivingCycles.tcl. There are certain restrictions of this gear selection procedure, which is defined in [8]. Above all, the implementation of the gear selection and shift point determination are ONLY for vehicles equipped with manual transmissions and conventional combustion engines.

Required data and precalculations

In each second of the driving cycle, the velocity is given per regulation. The vehicle should follow the speed profile. It has to be calculated how much acceleration or deceleration, and which gear is needed to reach the velocity in the next second. In a vehicles view it should be considered that, for the velocity in next second, how much power does it require to reach it. Furthermore, how much power is available in each gear? How high is then the engine speed under each gear?

The first selection of the gear will be, when in this second more than one gear fulfil the requirement of power, the highest possible gear should be selected. Therewith the engine works in a low speed.

To know clearly, which gears are available in a certain second, the following data in Table 4.12 are required and calculations shall be performed, in order to determine the gears to be used. It is mostly done in the proc Generic_precalc and proc WLTC_Pre_Calc in tcl File (except n_min_drive because of its speed trace dependence, it will be calculated in proc WLTC_Gear_Determination).

Variables in Tcl File	Units	Description
p_rated	W	rated engine power
n_rated	min^{-1}	rated engine speed
n_idle	min^{-1}	engine idle speed
n_g	-	number of gears
n_dv	-	(engine speed in rpm)/(vehicle speed in km/h) = n_dv (for each gear)
f_0	-	constant road load coefficient
f_1	-	first order road load coefficient
f_2	-	Second order road load coefficient
k_r_amp	-	amplify factor for k_r to modify the calculation of required power in case of acc.
k_r	-	factor related to the calculation of power required in acceleration
n_max	-	max {n_max_ng,n_max_95}, rpm
n_max_ng	min^{-1}	the engine speed in which the vehicle reached its maximum speed with the gear ng_vmax
n_max_95	min^{-1}	the minimum engine speed where 95 percent of rated power is reached
n_min_drive	min^{-1}	the minimum engine speed for each gear when the vehicle is in motion
TM	kg	test mass of the vehicle
pmr	W/Kg	power mass ratio
ng_vmax	-	the gear in which the maximun vehicle speed is reached
rpm	min^{-1}	the engine speeds in engine look-up table
pow	W	the engine full load power in the whole engine speed range
v_max	Km/h	max. vehicle speed
SM	-	safty margin
ASM	-	additional safty margin if required by manufacturer

Table 4.12: Required data and precalculations for gear selection

The required data **TM, n_g, n_dv** and **n_idle** can be directly read from the Car Parameters using IFileRead command or calculated from the relevant car parameters.

The rated engine power **p_rated** is provided by the manufacturer. When an explicit value is not given, it can be approximated based on the engine look-up table. The biggest power calculated out of all full load data points is the rated engine power. The engine speed at which the rated engine power at first reached is the rated engine speed **n_rated.**

pmr: Power-mass ratio = rated engine power (W)/TM(kg), used as classification criteria.

pow: the full load engine power in the whole engine speed range, calculated from the engine speed-torque map namely the full load power look-up table. The available power of each gear in every second will be approximated by a linear interpolation.

v_max: the maximum vehicle speed. When the manufacturer does not provide it, we set the corresponding value that we get from the calculation of ng_vmax (see below) as the maximum vehicle speed.

The road load coefficients f_**0, f_1** and **f_2**: As described in [8], as an alternative for determining road load with the coastdown or torque meter method, a calculation method for default road load can be used. In this application the default road load will be used.

For the calculation of a default road load based on vehicle parameters, several parameters such as test mass, width and height of the vehicle shall be used. The default road load shall be calculated for the reference speed points.

The default road load force shall be calculated using the following equation:

$$Fc = f\_0 + f\_1 * v + f\_2 * v^2$$

where:

Fc: is the calculated default road load force as a function of vehicle velocity, N;

f_0: is the constant road load coefficient, in N, defined by the following equation:

$$f\_0 = 0.14 * TM;$$

f_1: is the first order road load coefficient and shall be equal set to zero;

f_2: is the second order road load coefficient, in $N \cdot (h/km)^2$, defined by the following equation:

$$f\_2 = (2.8 * 10^{-6} * TM) + (0.017 * Area);$$

v: is vehicle velocity, km/h;

TM: test mass, kg;

Area: vehicle width* vehicle height, which can be directly read from Car Parameters.

k_r: is a factor taking the inertial resistances of the drivetrain during acceleration into account and is set to 1.03

k_r_amp: In some cases the calculated required power seems to be smaller than it actually needed to acceleration. It means that a higher gear will be selected too early. In such cases we set an amplify factor to modify the calculation of required power.

ng_vmax:, the gear in which the maximum vehicle speed is reached and shall be determined as follows:

If $v_{max}(ng) \geq v_{max}(n\_g-1)$, then,

ng_vmax = n_g

otherwise, ng_vmax = n_g -1

where:

vmax(ng) is the vehicle speed at which the required road load power equals the available power in gear n_g (see Figure 4.32).

vmax(ng-1) is the vehicle speed at which the required road load power equals the available power, $P_{wot,}$ in the next lower gear (see Figure 4.33).

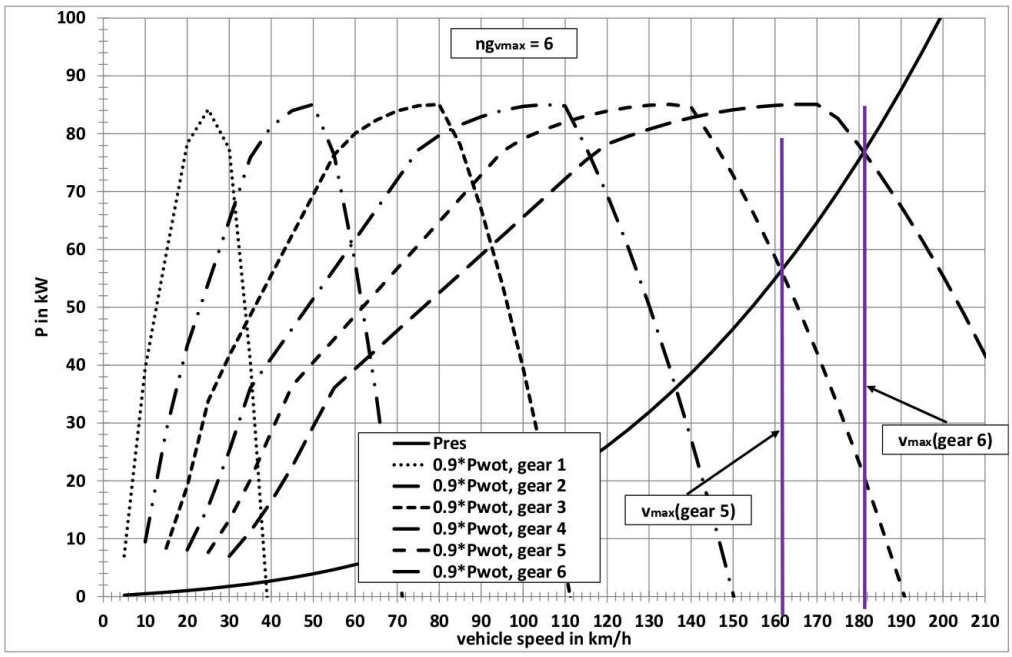


Figure 4.32: An example where ng_vmax is the highest gear [8]

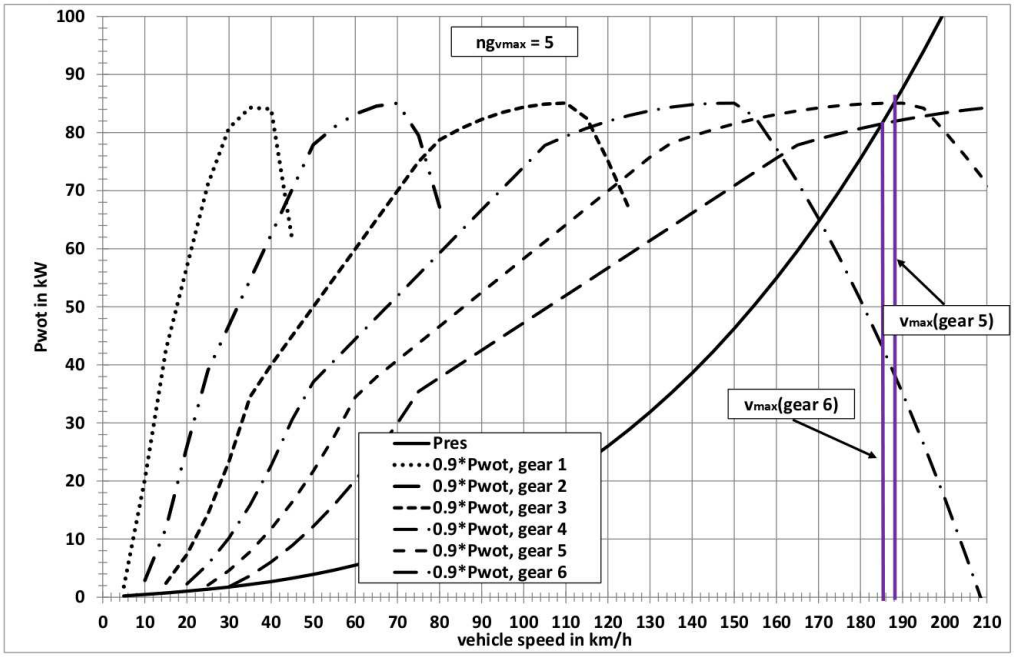


Figure 4.33: An example where ng_vmax is the 2nd highest gear [8]

Definition of **n_min_drive**, the minimum engine speed when the vehicle is in motion, min^{-1}.

For n_gear = 1,

n_min_drive = n_idle,

For n_gear = 2,

(a) For transitions from first to second gear during accelerations from standstill:

$$n\_min\_drive = 1.15 * n\_idle,$$

(b) For decelerations to standstill:

$$n\_min\_drive = n\_idle.$$

(c) For all other driving conditions:

$$n\_min\_drive = 0.9 * n\_idle.$$

For n_gear > 2, n_min_drive shall be determined by:

$$n\_min\_drive = n\_idle + 0.125 (n\_rated - n\_idle).$$

Downscaling

The cycle to be driven depends on the test vehicle's rated power to mass (in running order) ratio [in W/kg], and its maximum velocity, vmax [in km/h]. Driveability problems may occur for vehicles with power to mass ratios close to the borderlines between Class 1 and Class 2, Class 2 and Class 3 vehicles, or very low powered vehicles in Class 1. Since these problems are related mainly to cycle phases with a combination of high vehicle speed and high accelerations rather than to the maximum speed of the cycle, the downscaling procedure shall be applied to improve driveability. In the tcl-file the procedure is implemented in **proc WLTC_Downscaling**.

Determination of the downscaling factor

The downscaling factor, **f_dsc**, is a function of the ratio, **r_max**, between the maximum required power of the cycle phases where the downscaling is to be applied and the rated power of the vehicle, **p_rated**.

The maximum required power, **p_req,max,i** [in kW], is related to a specific time i and the corresponding vehicle speed **v_i** in the cycle trace and is calculated using the following equation:

$$p\_req,max,I = (f\_0 \ v\_i) + (f\_1 \ v\_i2) + (f2 \ v\_i3) + (1.03 \ TM \ v\_i \ a\_i)$$

r_max shall be calculated using the following equation:

$$r\_max = p\_req,max,I / p\_rated$$

The specific time i for class 1, 2 and 3 is different and is given in [8].

The downscaling factor, **f_dsc**, shall be calculated using the following equations:

If r_max < r0, then f_dsc = 0 and no downscaling shall be applied. If r_max ≥r0, then f_dsc = a_1 r_max + b_1. The constants **r_0**, the factors **a_1** and **b_1** are different for class 1, 2 and 3. They are given in [8].

The resulting **f_dsc** is mathematically rounded to three decimal places and is only applied if it exceeds 0.01.

Figure 4.34 shows an example for a downscaled extra high speed phase of the Class 3 WLTC.

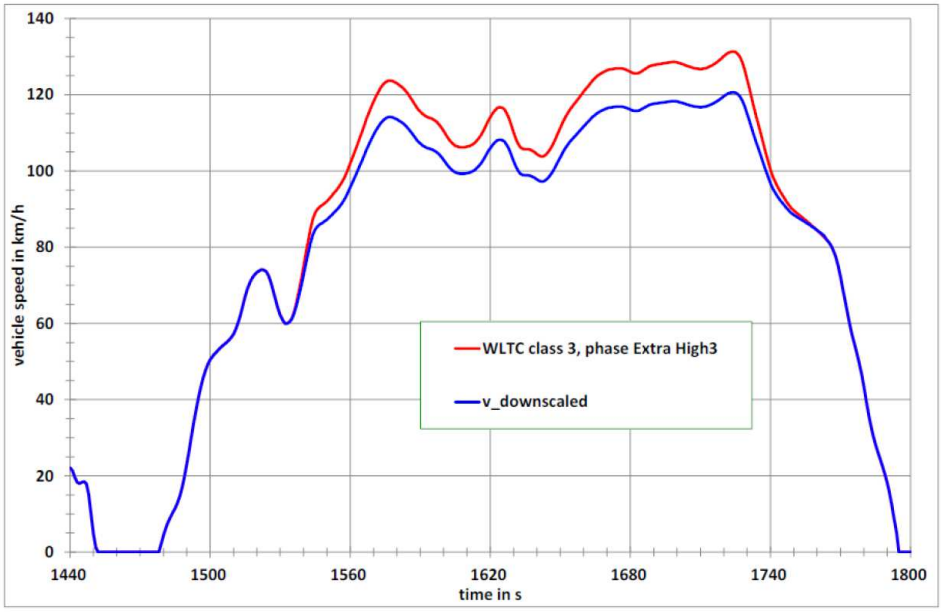


Figure 4.34: Downscaled extra high speed phase of the Class 3 WLTC [8]

Calculations of possible gear to be used

This procedure is implemented in **proc WLTC_Gear_Determination**.

Calculation of required power

For every second j of the cycle trace, the power required to overcome driving resistance and to accelerate shall be calculated using the following equation:

$$p\_required,j = (f\_0 * v\_j + f\_1 * v\_j^2 + f2 * v\_j^3) + k\_r * a\_j * v\_j * TM$$

where:

p_required,j is the required power at second j, in [kW];

a_j is the vehicle acceleration at second j, in [m/s^2],

$$a\_j = (v\_(j+1) - v\_j) / [3.6 * (t\_(j+1) - t\_j)];$$

Determination of engine speeds

For any v_j ≤ km/h, it shall be assumed that the vehicle is standing still and the engine speed shall be set to n_idle. The gear lever shall be placed in neutral with the clutch engaged except 1 second before beginning an acceleration from standstill where first gear shall be selected with the clutch disengaged.

For each v_j ≥ 1 km/h of the cycle trace and each gear i, i = 1 to n_g, the engine speed, n_i,j, shall be calculated using the following equation:

$$n\_i,j = n\_dv\_i \;\; v\_jl$$

Selection of possible gears with respect to engine speed

The following gears may be selected for driving the speed trace at v_j:

(a) all gears i < ng_vmax where n_min_drive ≤ n_i,j ≤ nmax_95, and

(b) all gears i ≥ ng_vmax where n_min_drive ≤ ni,j ≤ n_max_ng

If a_j ≤ 0 and n_i,j drops below n_idle, n_i,j shall be set to n_idle and the clutch shall be disengaged.

If a_j > 0 and n_i,j drops below (1.15 n_idle), n_i,j shall be set to (1.15 n_idle) and the clutch shall be disengaged.

Calculation of available power

The available power for each possible gear i and each vehicle speed value of the cycle trace, v_i, shall be calculated using the following equation:

$$p\_available\_i,j = pow(n\_i,j)(1 - (SM+ASM))$$

Determination of possible gears to be used

The possible gears to be used shall be determined by the following conditions:

(a) The conditions of paragraph "Selection of possible gears with respect to engine speed" are fulfilled,

(b) p_available_i,j ≥ p_required,j

The initial gear to be used for each second j of the cycle trace is the highest final possible gear, i_max. When starting from standstill, only the first gear shall be used.

Additional requirements for corrections or modifications of gear use

The initial gear selection shall be checked and modified in order to avoid too frequent gear-shifts and to ensure driveability and practicality.

Corrections and/or modifications shall be implemented in tcl File as below:

proc *Gear_Mod_Acc_NoBackwards*

proc *Gear_Mod_Acc_NoSkip*

proc *Gear_Mod_Dec_Dur_Hdl*

proc *Gear_Mod_Dec_Skip_OneNull*

proc *Gear_Mod_Dec_2thGear*

proc *Gear_Mod_Sequence_Scan*

In the following the procedures listed above will be shortly explained.

proc *Gear_Mod_Acc_NoBackwards*

If a lower gear is required at a higher vehicle speed during an acceleration phase, the higher gears before shall be corrected to the lower gear.

Example: During an acceleration phase the original calculated gear used is 2, 3, 3, 3, 2, 2, 3. In this case the gear use shall be corrected to 2, 2, 2, 2, 2, 2, 3.

proc *Gear_Mod_Acc_NoSkip*

Gears used during accelerations shall be used for a period of at least 2 seconds (e.g. a gear sequence 1, 2, 3, 3, 3, 3, 3 shall be replaced by 1, 1, 2, 2, 3, 3, 3). Gears shall not be skipped during acceleration phases.

proc *Gear_Mod_Dec_Dur_Hdl*

During a deceleration phase, gears with n_gear > 2 shall be used as long as the engine speed does not drop below n_min_drive.

The second gear shall be used during a deceleration phase within a short trip of the cycle as long as the engine speed does not drop below (0.9 * n_idle).

If the engine speed drops below n_idle, the clutch shall be disengaged.

proc *Gear_Mod_Dec_Skip_OneNull*

If the duration of a gear sequence is only 1 second, it shall be replaced by gear 0 and the clutch shall be disengaged.

If the duration of a gear sequence is 2 seconds, it shall be replaced by gear 0 for the 1st second and for the 2nd second with the gear that follows after the 2 second period. The clutch shall be disengaged for the 1st second.

Example: A gear sequence 5, 4, 4, 2 shall be replaced by 5, 0, 2, 2.

proc *Gear_Mod_Dec_2thGear*

If the deceleration phase is the last part of a short trip shortly before a stop phase and the second gear would only be used for up to two seconds, the gear shall be set to 0 and the clutch may be either disengaged or the gear lever placed in neutral and the clutch left engaged.

A downshift to first gear is not permitted during those deceleration phases.

proc *Gear_Mod_Sequence_Scan*

If gear i is used for a time sequence of 1 to 5 seconds and the gear prior to this sequence is lower and the gear after this sequence is the same as or lower than the gear before this sequence, the gear for the sequence shall be corrected to the gear before the sequence.

Examples:

(i) gear sequence i-1, i, i-1 shall be replaced by i-1, i-1,i-1;

(ii) gear sequence i-1, i, i, i-1 shall be replaced by i-1, i-1, i-1, i-1;

(iii) gear sequence i-1, i, i,i, i-1 shall be replaced by i-1, i-1,i-1, i-1, i-1;

(iv) gear sequence i-1, i, i, i, i, i-1 shall be replaced by i-1, i-1, i-1, i-1, i-1, i-1;

(v) gear sequence i-1, i, i, i, i, i, i-1 shall be replaced by i-1, i-1, i-1, i-1, i-1, i-1, i-1.

proc *Gear_Mod_Acc_NoBackwards* to **proc** *Gear_Mod_Sequence_Scan* shall be applied sequentially, scanning the complete cycle trace in each case. Since modifications may create new gear use sequences, these new gear sequences shall be checked three times and modified if necessary.

After these modifications and corrections, we implement the gear strategy in CarMaker to evaluate the performance. Figure 4.35 shows a section of the driving cycle performance with DemoCar. We can see that there are a problem in acceleration and deceleration phase.

Settings: min. duration of acceleration 2s, neutral gear transition between skip in deceleration.

Figure 4.35: Gear selection without optimization (vehicle: DemoCar_DrivingCycles)

As it turns out, two additional or optional modifications can be helpful to improve the actual velocity trace with the result that the velocity stays in the limits:

proc *Gear_Mod_Acc_Dur_3s*

proc *Gear_Mod_Dec_Skip_NoNull*

proc *Gear_Mod_Acc_Dur_3s*

As we can see in Figure 4.35, the upwards gear changes during the acceleration are combined with a transient backward-going velocity due to the execution of clutch. A too often gear changes may lead to the out-of-range of car velocity. Therefore, we suggest a minimum duration of 3 seconds for each gear during the acceleration. Figure 4.36 shows the results of the same section after implementing this suggestion.

Figure 4.36: Gear selection with 1st optimization (vehicle: DemoCar_DrivingCycles)

proc *Gear_Mod_Dec_Skip_NoNull*

As we can see in Figure 4.35, the gear sequence in 279-283s is originally 4-0-2-2-2. However, the actual gear sequence in powertrain is 4-0-1-1-1. From the 4th gear to transient neutral, the clutch stays quasi disengaged, the engine speed reduced to idle speed. After that, despite the setting back of Driver.SetGearNo to 2nd gear, the actual gear in powertrain change to and then stays in 1st gear. In this process, the car keeps braking and finally goes out of the velocity tolerance range. That is to say, CarMaker with its own logic does not react as expected, when it comes to the neutral gear transition between gear skips in deceleration. Without neutral gear transition, as shown in Figure 4.37, the gear sequences follows as expected and the car velocity does not go out of range.

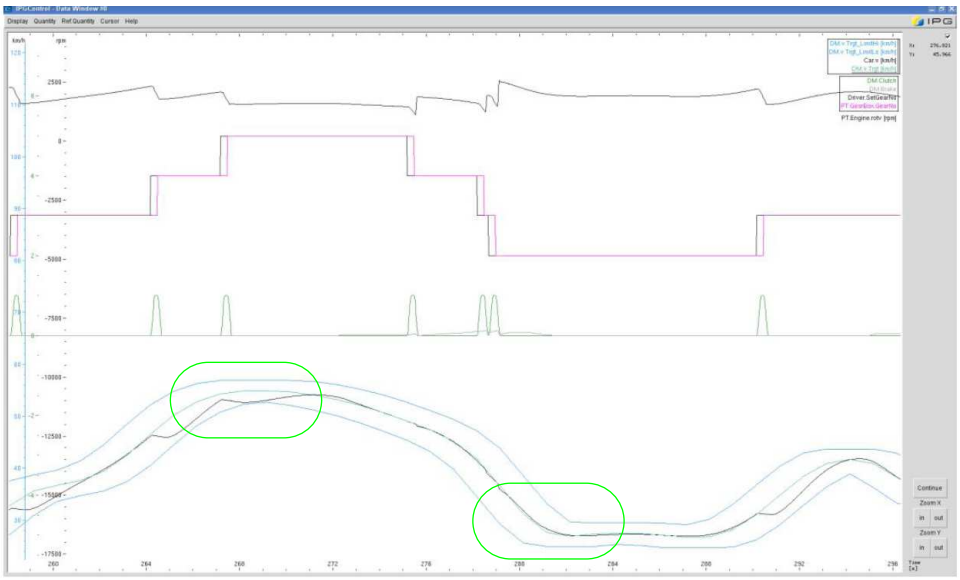


Figure 4.37: Gear selection with 2nd optimization (vehicle: DemoCar_DrivingCycles)

Since the behavior in driving cycles with specific gear-strategy is from car to car differently, the **proc** *Gear_Mod_Acc_Dur_3s* and **proc** *Gear_Mod_Dec_Skip_NoNull* are optional in the whole procedure and can be deactivated easily by uncommenting the corresponding command with "#" in **proc** *Gear_Selection*. It should be mentioned that only one of the two modification procedures **Gear_Mod_Dec_Skip_OneNull** and **Gear_Mod_Dec_Skip_NoNull** could be chosen at a time.

4.2 DrivingScenarios

AMS_TestTrack

This TestRun shows the "auto-motor und sport" (AMS) fuel consumption round in the Stuttgart area. AMS is a well-known german car consumer magazine. The AMS_Testtrack is used to examine the real-life fuel consumption or range of tested cars.

The length of one round is about 92 km. In reality, the journalists drive the route three times to be more independent from single effects like traffic jam or slow cars on the road. All speed limits, traffic signs and traffic lights, which are on this track are included in the TestRun. The stopping-time in front of a stop sign is set to 5 seconds. You can observe the route and the vehicle via GoogleEarth (File >Connect GoogleEarth in IPGMovie).

Figure 4.38: View of the AMS lap in IPGMovie and Google Earth

BerninaPass

This TestRun is an uphill driving scenario on the Bernina Pass in Switzerland, which is famous for Powertrain testing.

The TestRun shows:

- scenario for powertrain testing
- connection to Google Earth

The route starts in Pontresina at 1800 m above sea level and ends at the highest point of the pass at an altitude of 2328 m. There are some sharp curves on the way to the top, the speed limit is 80 km/h. You can observe the route and the vehicle via GoogleEarth by using the feature *IPGMovie > File > Connect GoogleEarth* (in IPGMovie).

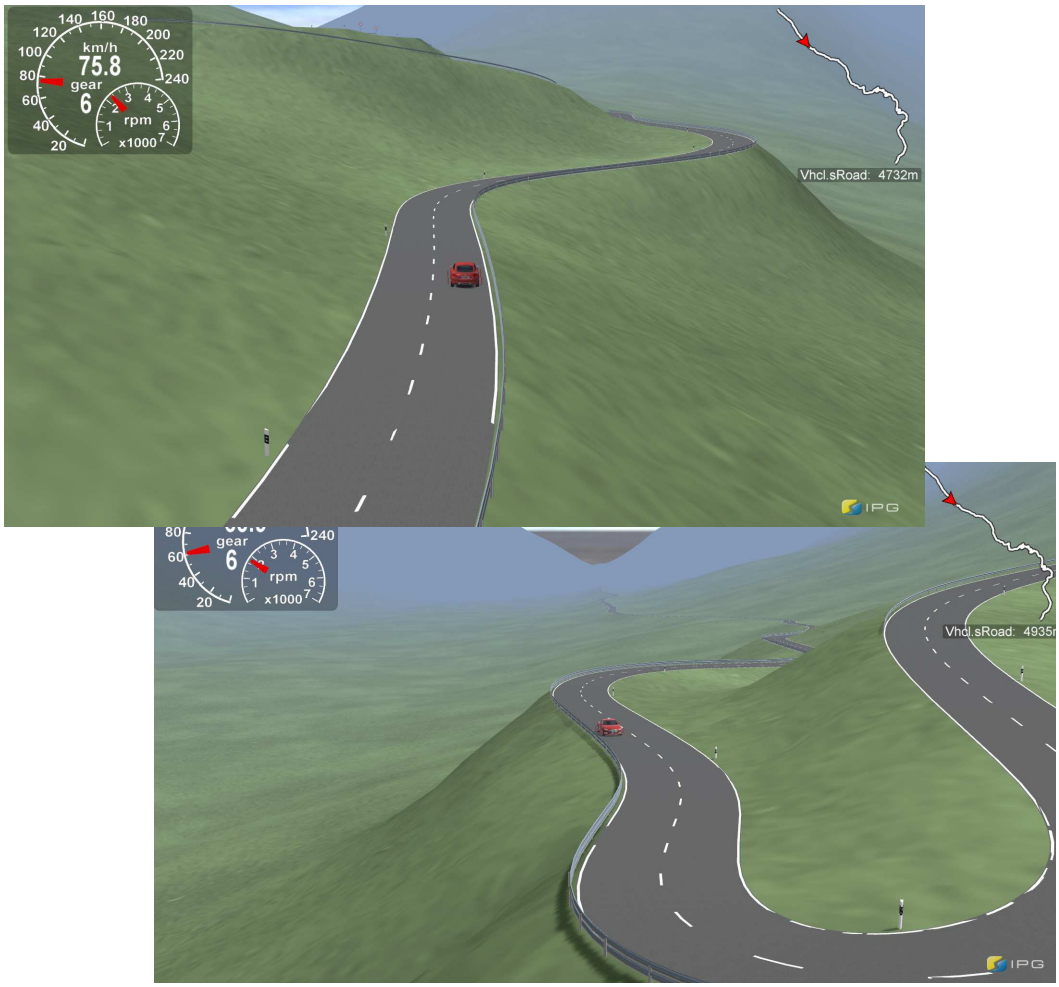


Figure 4.39: The Bernina Pass in IPGMovie

CityDriving

In this TestRun the ego-vehicle drives through an urban environment. The car is a Honda Fit with a CVT-transmission in the "single track" mode, which means the TCU tries to set the engine speed with lowest fuel consumption.

Figure 4.40: Urban city of example TestRun "CityDriving"

Karlsruhe_RDE_IPG_FreeDriving

The following TestRuns include a test track which corresponds to the real driving emissions (RDE) test regulations (EU2016/427, EU2016/646).

Figure 4.41 illustrates the course of the IPG RDE KA Route. The route follows the test sequence suggestion of the regulation. It starts at the IPG Automotive headquarter and leads into the inner city area of Karlsruhe, followed by driving through some rural environment combined with up- and downhill sections.The last part of the route is on the Autobahn (Highway).

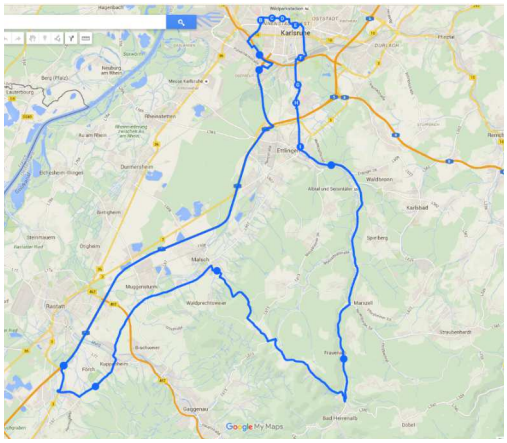


Figure 4.41: RDE IPG Karlsruhe test track

The TestRun Karlsruhe_RDE_IPG_FreeDriving is the test track with speed limits and traffic lights. The IPGDriver reacts on these boundary conditions, but within these limitations he can act freely, according to his driving characteristics.

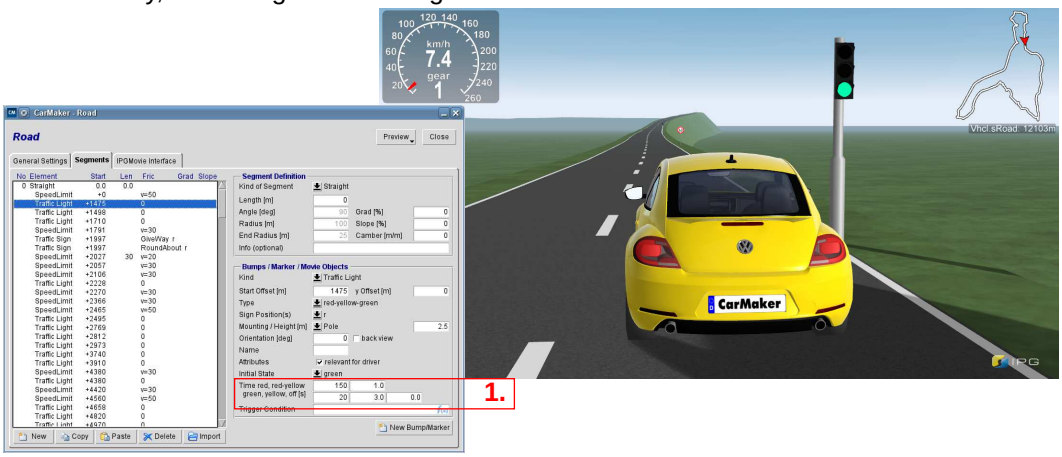


Figure 4.42: RDE TestRun in free driving mode

The road infrastructure (traffic lights, speed limits, stop signs, give away signs, round about traffic and urban area begin/end) is set in the *Main GUI > Parameters > Road* user interface (see Figure 4.42 on the left). The traffic lights are defined with different time controllers according to the experience gained from the test drive. The time is set in the GUI, see Figure 4.42 (1.).

In order to examine the influence of the driving characteristics to e.g. fuel consumption, there are different ways to adjust the behavior of the IPGDriver. The standard settings can be found in the *Main GUI > Parameters > Driver > Standard Parameters*.

Please be aware that there is partly no speed limit on the Autobahn, so the cruising speed should be lower than 145 km/h, if you want to stay within the RDE regulations.

In this example, the parameters **Long.AccuracyCoef** and **Long.SmoothCoef** are set to show some further possible variations beside the standard settings (*Main GUI > Parameters > Driver > Misc./Additional Parameters*).

Further information about influencing the throttle/brake are shown in the IPGDriver Manual chapter Use Case: Influencing Throttle/Brake.

Nardo_HandlingTrack

The TestRun shows a handling track referring to the Nardo Handling track. The track shows banked curves, uphill as well as down hill sections.

The TestRun shows the following features:

- segment based racing circuit
- performance of the racing driver

The track is built out of consecutive road segments defined in the road user interface, see *MainGUI > Parameters > Road > Segments.* The segments are defined with length, radius and other properties for defining the lateral slope or the gradient. Further information about the road segments are described in the CarMaker User's Guide chapter *Road segment.*

Figure 4.43: Demo_Audi_R8 on the handling track

The so called *Racing Driver* is selected via the graphical user interface of the driver, see *MainGUI > Parameters > Driver.* For further information, please read CarMaker Driver Manual chapter *Race Driver*.

OffroadProvingGround

This TestRun represents a proving ground for all wheel driven vehicles with a high ground clearance and off-road capability. The vehicle drives one round. On the way are several challenges, like slopes, cones or beams, the vehicle has to pass.

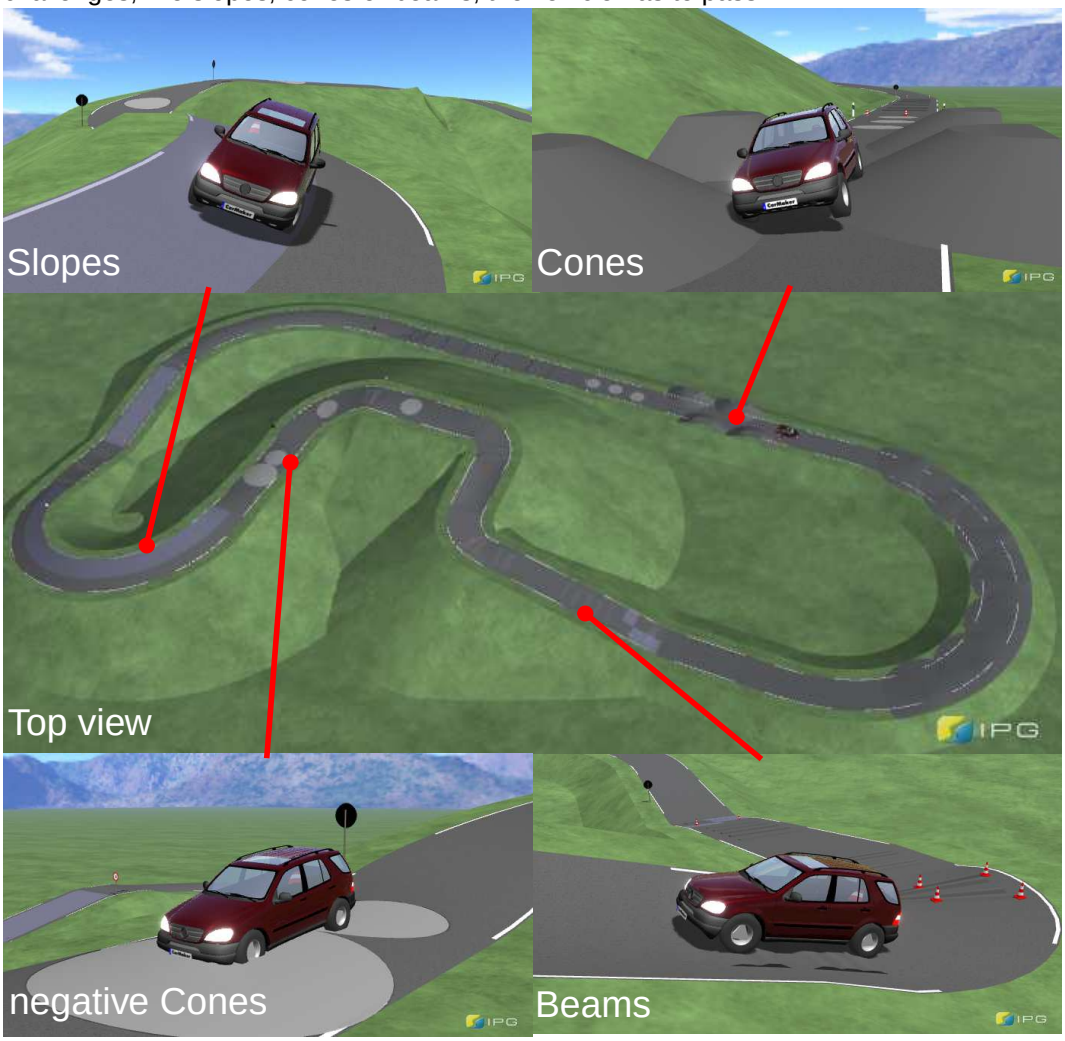


Figure 4.44: Offroad proving ground overview

RacePitStopRefuel

This TestRun shows a pit stop scenario. The pit stop is controlled by the absolute fuel consumption and the driven distance of the vehicle.

The TestRun shows the following features:

- fuel consumption of the combustion engine model
- jumping between maneuvers
- the DVA mechanism, which allows you to directly manipulate quantities states during the simulation

The vehicle drives on the closed track named "Hockenheim" and is controlled by different maneuvers. With the first maneuver (No. 0) the car drives until it reaches the *End Condition*

```
1:  PT.Control.Consump.Fuel.Abs>(1.3*Qu::PitStopCount) && Car.Road.sRoad>2000 &&
    Car.Road.sRoad<2130
```

If the absolute fuel consumption is greater than 1.3*Qu::PitStopCount (liter) and the driven distance is greater than 2000 m and lower than 2130 m, the car will proceed with entering the pit lane (pit lane process Man. no. 1 - 8).

In maneuver no. 1 the driver will slow down the velocity. The maneuver ends after reaching Car.Road.sRoad >2180 m (absolute distance of the road).

In maneuver no. 2 the driver performs a lateral *Track Offset (-15 m).* The car drives for 100 m with 80 kph through the pit lane and stops with a defined deceleration of -8 m/s^2 , see maneuver no. 3 and no. 4. The defined *Track Offset* is still kept at -15 m.

The refuel is controlled by the quantity PT.Engine.FuelFlow_ext, which is set by DVA command to perform a 1 liter per second refilling maneuver. The pit stop counter is increased by one.

```
DVAwr PT.Engine.FuelFlow_ext Abs 5000 -1
Eval mfirst () ? Qu::PitStopCount++
```

When the car starts driving out of the pit lane the refilling is stopped by the command.

```
DVAwr PT.Engine.FuelFlow_ext Abs -1 0
```

The car drives back on the track by setting the track offset to zero, see maneuver no. 7. In maneuver no. 8 the command

```
Eval ManJump drive
```

sets the maneuver selection back to the first maneuver labeled with "drive". Now it starts over again until the lap number is greater than three, see minimaneuver command in maneuver no. 0.

```
Eval DM.Lap.No>3 ? ManJump(9)
```

The fuel consumption is defined in the vehicle data set > Powertrain > Drive Source > Engine > Fuel Consumption. For further information see CarMaker User's Guide chapter *Parametrizing the Engine.*

StelvioPass_withTrailer

This TestRun is an uphill driving scenario on one part of the Stilfser Joch in Italy.

The TestRun shows the functions:

- driving with trailer
- additional load
- connection to Google Earth

The route starts at an altitude of 1300 m above sea level, the last point of this TestRun is 580 m higher. The vehicle pulls a horse trailer with an additional loading of 200 kg defined in the *Main GUI > Load > Trailer Loads* interface. There are a lot of sharp curves on the way upwards. You can observe the route and the vehicle via GoogleEarth by using the feature *IPGMovie > File > Connect GoogleEarth* (in IPGMovie).

4.3 ElectricAndHybrids

ElectricalDriveRecharge

The intergrated battery model can be used to simulate the vehicle's charging state. This TestRun demonstrates how to:

- define additional electrical loads,
- simulate charging stations.

In this TestRun an electrical vehicle is used. The charging state of the vehicle's battery is low.

Using a DVA command an additional electrical load (air conditioning) is activated. The additional load created a power demand of 4.5kW for the first 20 seconds of simulation. See *MainGUI > Parameters > Maneuver > Minimaneuver Commands*.

```
1:  #Air condition 4.5kw for the first 20sec, after that comes down to 1.5kw
2:  Eval Time<=20  ? (PT.PwrSupply.HV1.Pwr_aux=4500)
```

After 20 seconds, the power demand of the air conditioning is decreased to 1.5kW, see line 2 and 3. In lines 4 and 5 additional loads are activated.

```
3:  Eval Time>20 && PT.PwrSupply.HV1.Pwr_aux>=1500 ? PT.PwrSupply.HV1.Pwr_aux = 4500 +
    (20-Time*60)
4:  Eval first(PT.PwrSupply.HV1.Pwr_aux<=1500) ? PT.PwrSupply.HV1.Pwr_aux = 1500
```

If the charging level of the battery falls below 7 percent the warning "Low Battery Level" is displayed in the Session Log (MainGUI > Simulation > Session Log).

```
5:  #Error as soon as battery is empty
6:  Eval first (PT.BCU.SOC_HV <= 7) ? LogWarn("Low Battery Level")
```

The overlay in the IGPMovie window is configured by a script file, see "installation directory > */Movie/3D/Utilities/ConsumersEnergyOverlay.tclgeo*". If you want to modify it, make a copy to your local "Movie" folder in your project directory and modify it accordingly using a text editor. To integrate the overlay use *MainGUI > Parameters > Road > IPGMovie Interface > User -defined Background*.

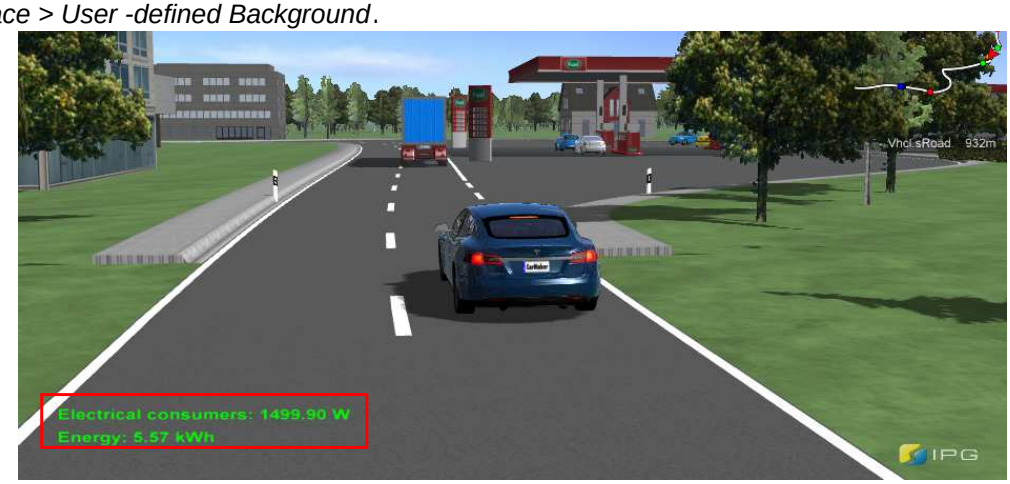


Figure 4.45: IPGMovie information overlay, defined by User-defined Background

Before the vehicle makes a turn to the petrol station the indicators are activated:

```
7:  Eval first(Car.Road.sRoad > 810) ? DM.Lights.Indicator = -1
8:  Eval first(Car.Road.sRoad > 960) ? DM.Lights.Indicator = 0
```

Charging the battery is simulated using a DVA command. After the vehicle has been turned off (Vhcl.OperationState=0) the battery's charging level is increased by 0.002 Ah every millisecond for a duration of 20 seconds.

```
 9:  VhclOp=absent
10:  [Vhcl.OperationState ==0] DVAwr PT.BattHV.AOC Off 20000 0.002
```

When the charging maneuver is finished the vehicle leaves its parking space to go on driving on the country road.

HybridAxleSplitDrive

A vehicle with a hybrid axle-split powertrain drives through different areas. Watch the different operation strategies (View > Overlay-Left > Speed and Powertrain) for the powertrain. In the urban area the "electric driving" mode is mainly used, whereas in the mountain and highway area, the combustion engine has to jump in.

Figure 4.46: Using a vehicle with a hybrid split axle drivetrain

HybridParallelP2MountainDrive

A P2-Hybrid driving up- and downhill the Bernina Pass, from Pontresina to Poschavio. The different operating strategies like "engine drive", "electric drive" or "regenerative brake" can be observed. On the way up, the battery is discharged, while it is recharged on the way down. At a certain State of Charge (SoC) the electric driving is blocked until the battery is recharged to defined value again.

4.4 PerformanceTests

AccelerationElasticity

To objectively evaluate the acceleration capabilities of a vehicle several tests can be performed. In these tests the time needed to accelerate the vehicle to a given velocity is measured.
The features of the TestRun are

- the definition of maneuver *End Conditions*,
- the usage of *RealTimeExpressions* for the creation of Quantities, Trigger conditions and the storage of the acceleration time as well as
- the manual control of the longitudinal dynamics.

To record the measured times quantities are defined. These quantities can also be monitored using IPGControl.

```
Eval first() ? Qu::Time_2_100kph=0
```

As the vehicle does not immediately begin to move after the TestRun has been started, the starting time has to be evaluated. To do so the trigger condition *"Car.v* > 0.001*"* is applied.

```
Eval first (Car.v > 0.001) ? Qu::Start_Time=Time
```

As the vehicle reaches a defined velocity the respective acceleration time is evaluated by subtracting the starting time from the current time.

```
Eval first (Car.v >= 27.77) ? Time_2_100kph = (Time - Start_Time)
```

The tests of acceleration capability are performed with an initial velocity of zero (see Table 4.13). The different measures are evaluated in the acceleration maneuver.

Acceleration				
0-80 km/h	0-100 km/h	0-130 km/h	0-160 km/h	0-180 km/h

Table 4.13: Acceleration tests from stand still

In the second maneuver, the vehicle is decelerated to a velocity of 60 kph. Additionally, the acceleration times are displayed in the *Session Log.*

```
Eval first () ? Log ("0-100 km/h : %.2f s", Time_2_100kph)
```

The elasticity describes the time a vehicle needs to accelerate from a given starting velocity to another given terminal velocity. For these tests a certain gear selection is mandatory (see Table 4.14).

Elasticity	
60-100 km/h (in second last gear)	80-120km/h (in last gear)

Table 4.14: Acceleration elasticity tests

When the vehicle has reached to defined starting velocity (60 kph) a manual maneuver is applied to fix the gear and accelerate the vehicle: *Maneuver* > *Longitudinal Dynamics* > *Manual (Pedals, Gear).* At the begin of the elasticity measurement the starting time (absolute TestRun time) is measured. This measurement is repeated for the second elasticity characteristic.

Attention! The end velocities for the acceleration tests should not exceed the maximum vehicle speed! In this case, adapt the TestRun maneuver to reduce the test velocities.

CO2Emission

This TestRun demonstrates a section of the WLTC test cycle.

It demonstrates the following features:

- Speed Profile
- "Session Log" output

In the TestRun the vehicle follows the speed profile "WLTC class 3". This speed profile is selected by clicking on MainGUI > Parameters > Input from File and activating the CarMaker quantities to be overwritten.

To evaluate the vehicle's fuel consumption and to calculate the emissions exhausted, respectively, a "Fuel Consumption Map" has to be defined in the vehicle's parameters, see Fuel Consumption Map.

If the TestRun has finished or is paused, the Session Log is opened and the previously calculated values of the vehicle's fuel consumption and CO2 emission are shown. This function is accomplished by the minimaneuver commands of the second maneuver. This maneuver has a duration of "0" (Final Maneuver), see User's Guide chapter *Final Minimaneuver.* It is executed whenever the TestRun is stopped or has been finished. The following lines write to the Session Log:

```
Eval first() ? Log("CO2 Emission:")
Eval first() ? Log("Average fuel consumption: %.3f l/100km",PT.Control.Consump.Fuel.Avg)
Eval first() ? Log("Average CO2 emission: %.3f g/km",PT.Control.Consump.Fuel.Avg*23.7)
```

HillStarting

In this TestRun a vehicle coupled with a trailer is starting on a hill. It is possible to run the test without the trailer.

The TestRun's contents are

- starting on a hill with a trailer

The road used in this TestRun is put together of different segments with different ascents. The friction coefficient on the right side of the road is fixed. The friction coefficient on the left side on the road is reduced, see *MainGUI > Parameters > Road > Segments > Override selected Segment Attributes.*

The vehicle is forced to stop using an invisible Stop sign, see *MainGUI > Road > Segments > Stop*. Starting is done using the IPGDriver. During the execution of the TestRun the vehicle's motion is supervised. If the vehicle should move more than five meters backwards, the TestRun is stopped. To accomplish this end condition the maximum distance covered is recorded in the *Minimaneuver Commands* (see line 1). If the vehicle reaches its maximum climbing ability, it will roll back after each try to start and trigger the end condition (5m backwards). If the vehicle is able to climb the entire track, maneuver number 2 is started, see line 2.

```
1:  Eval s_max=max(s_max, Vhcl.sRoad)
2:  Eval first (s>1125) ? ManJump ("2")
```

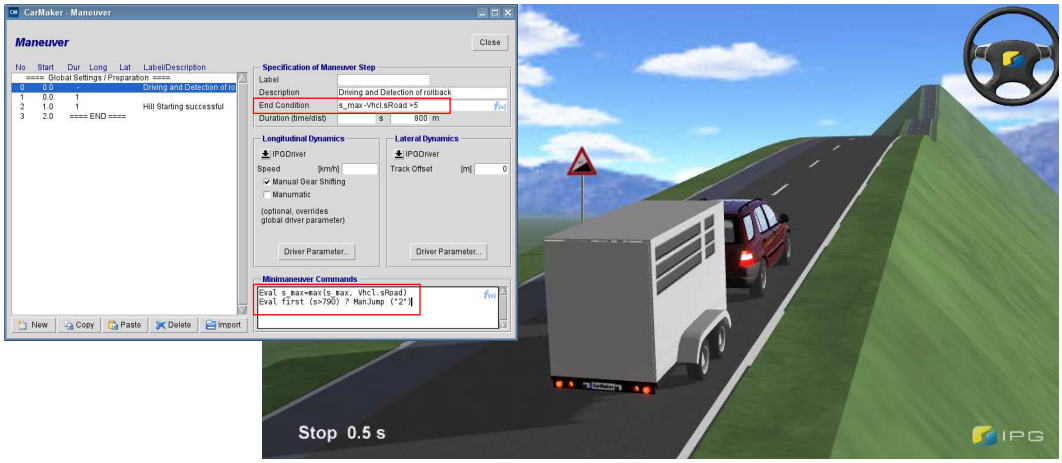


Figure 4.47: Hill starting Test with Trailer

LapTime

This TestRun is similar to the TestRun **Racetrack_Hockenheim.** It shows the short race circuit Hockenheim in Germany.

The Track is made out of Segments. And the corners are prepared with curbs. The defined driver is the Racing Driver, see *MainGUI > Parameters > Driver.*

As long as the Race Driver is used, lap times will be written to the Session Log automatically. If the user parameterized driver is used, lap times can be shown in IPGControl using the quantity **DM.Lap.Time**. Additionally, several sectors of the track can be defined, see *MainGUI > Parameters > Road > Segments > Trigger Point.* The time taken in each sector can be displayed in IPGControl using the quantities **DM.TriggerPoint.Time** and **DM.TriggerPoint.Id**.

In this TestRun, the times taken in each sector one to three are written to the Session Log each round using the minimaneuver commands below. Lines 1 to 3 record the times whenever the current sector changes differentiating the different sectors' Trigger ids.

```
1:  Eval change(DM.TriggerPoint.Id) ? (DM.TriggerPoint.Id==1) ?
    Sector1=DM.TriggerPoint.Time;
2:  Eval change(DM.TriggerPoint.Id) ? (DM.TriggerPoint.Id==2) ?
    Sector2=DM.TriggerPoint.Time;
3:  Eval change(DM.TriggerPoint.Id) ? (DM.TriggerPoint.Id==3) ?
    Sector3=DM.TriggerPoint.Time;
```

The following lines write the sector time to the Session Log:

```
4:  Eval change(DM.Lap.No) ? Log("1. Sector %.3f",Sector1)
5:  Eval change(DM.Lap.No) ? Log("2. Sector %.3f",Sector2)
6:  Eval change(DM.Lap.No) ? Log("3. Sector %.3f",Sector3)
```

Nardo_HighSpeedRing

This TestRun demonstrates a high speed testing track. The track consists of four lanes that are banked. The track enables a vehicle to drive at high speed for a long time. Because the track is banked only small steering angles (depending on the speed driven) are required.

This TestRun demonstrates:

- Driving on the high speed track on different lanes, as well as
- V-max test for one round including logging of the results

The lane to be driven on is adjusted by a "*Track Offset*" in the maneuver control, see *MainGUI > Parameters > Maneuver > Lateral Dynamics.* The values are to be interpreted with respect to the middle of the road, see *Road > General Settings > Driving Lane.* Due to the design of the track, see Figure 4.48, the left side of the track was removed, see *Road > General Settings > Global Road Attributes* (Track Width left=0).

If the track is driven on counter clock-wise, the *Track Offset* becomes negative. Thus the *Track Offset* has to be inverted if the driving direction is changed.

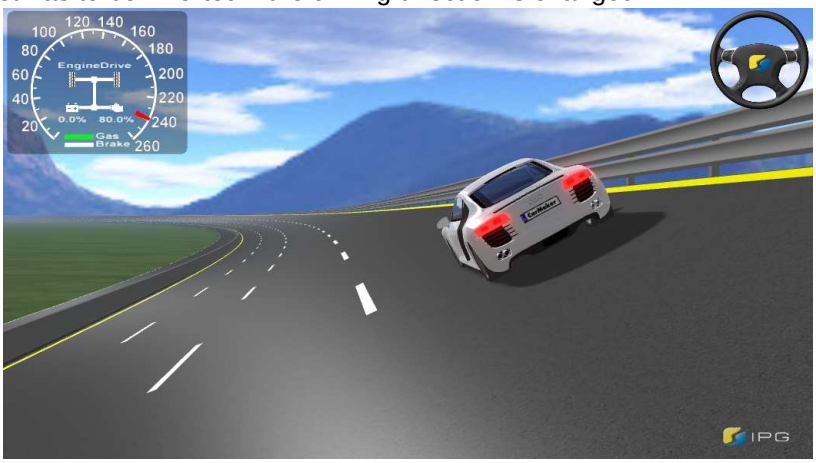


Figure 4.48: High Speed Ring with four lanes

The minimaneuver commands ensure output to the *Session Log*. The maximal velocity reached Vmax will be logged during the test due to the two lines of code below (see Global Settings /Preparation). The functions used there (first, max) are explained in the chapter *Operators and Functions* in the User's Guide.

```
1:  first () ? Qu::Vmax=0
2:  Vmax= max (Vmax, Car.v)
```

The vehicle drives on the track with its terminal velocity for one round. Then, the maximum velocity reached is logged to the *Session Log (Ctrl-L)*, see Maneuver ->Maneuver no. 7:*Minimaneuver Commands.*

```
1:  Eval first () ? Log ("Maximum velocity on the High Speed Ring:")
2:  Eval first () ? Log ("V-max: %.2f m/s",Vmax)
3:  Eval first () ? Log ("V-max: %.2f km/h",Vmax*3.6)
4:  Eval first () ? Log ("V-max: %.2f mph",Vmax*2.23694)
```

4.5 PowertrainControl

AdaptiveCruiseControl

Driving with AdaptiveCruiseControl on a typical german highway. A lot of traffic and no speed limit, so fast vehicles on the left lane, and lane changing vehicles on the middle and right lane.

Watch how the ACC controls the braking and accelerating.

It is activated by the mini maneuver command:

```
Eval first() ? AccelCtrl.ACC.IsActive=1
```

Then the desired speed is set to 150 km/h:

```
Eval first() ? AccelCtrl.ACC.DesiredSpd=150/3.6
```

And the ACC should control the vehicle, and not the IPGDriver:

```
Eval first() ? Driver.ReCon.Trf_Consider=0
```

And the IPGDriver should be set to considering the traffic.

Figure 4.49: Driving with longitudinal controller

Car2X_TrafficLight

A short TestRun, in which the car gets the information from a traffic light, how long the remaining red-phase will last. Depending on that information, the engine is stopped, or not. If the remaining time is under six seconds it remains running.

This example shows a simple logic and the principle how to use information from infrastructure, eg. a traffic light.

The commands are:

```
Eval nearTL = (Car.Road.sRoad>300 && Car.Road.sRoad<350)
Eval turns_green = (TrfLight.TL000.State==4)
Eval soon_green = (TrfLight.TL000.State==3 && TrfLight.TL000.tRemain<6)
Eval engine_on = (PT.Control.StrategyMode == 8)

Eval (nearTL && engine_on && (turns_green || soon_green)) ?
(PT.Control.StrategyMode_trg.SetManual=1;PT.Control.StrategyMode_trg=8):
PT.Control.StrategyMode_trg.SetManual=0
```

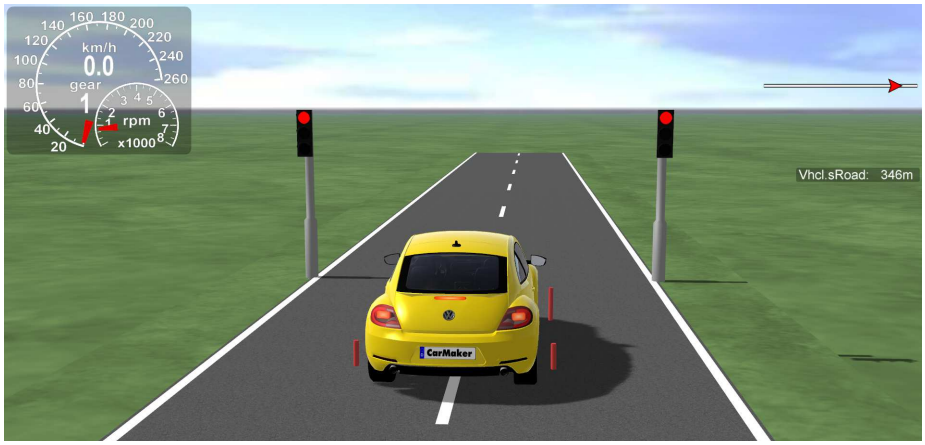


Figure 4.50: Interaction between traffic lights and the test car

PredictiveEnergyManagement

This example shows a fuel-saving operation strategy for the powertrain. With the help of the road sensor the vehicle knows a speed limit on the oncoming road and switches to "coasting" strategy.

This example is held very simple, to show the principles of the road sensor and how its information can be used.

In first maneuver, the driver drives the car until the sensor recognizes a speed limit in the distance of 250m (End condition: Sensor.Road.SpeedLimit.RMarker.Attrib.0!=-1)

After that the strategy mode "coasting" is enforced until the speed limit is reached:

```
Eval PT.Control.StrategyMode_trg.SetManual=1
Eval PT.Control.StrategyMode_trg=3
```

With the end condition: Car.v<speed_limit

The next preview function is, if the vehicle recognizes a downhill part of the road, after a grade, it forces the powertrain to electric drive, even if the SOC would not allow it. The logic behind that is, that the battery can be recharged on the downhill part. Of course, in reality it is far from being that simple, anyway this example should show how to handle with information from the road, and how to react.

Detect the downhill part: Sensor.Road.Grade.Path.LongSlope<-0.1

Enforce electric driving:

```
Eval PT.Control.StrategyMode_trg.SetManual=1
Eval PT.Control.StrategyMode_trg=4
```

Figure 4.51: Fuel saving strategy on an uphill track

TorqueVectoring

The TestRun shows a use case for rapid prototyping in the concept phase in power train architectures. It is shown how you can flexible adapt the torque distribution for different wheels by DVA command. For example torque vectoring concepts for testing the driveability under different conditions without having any controller model for the distribution of torque to the single wheels.

The TestRun includes the features:

- activate the *Coupling Model>Torque Vectoring* by *Vehicle Data Set > Powertrain > Driveline > (Front Axle/Rear Axle*/Diff. Center) to enable the corresponding variables for access the torque distribution ratio
- torque distribution ratio of the wheels
- showing the influence on the vehicles lateral dynamics

The quantities regulate the torque distribution between front and rear axle. And between left and right side of the vehicle. Further information about the coupling model are described in the User's Guide chapter *Parameterizing the Differential Coupling Model*. Information about the model and mathematical description are composed in the Reference Manual chapter *Coupling model "Torque Vectoring"*.

The TestRun shows three maneuvers performing an acceleration phase with different torque distributions. The torque distribution is set by manipulating the differential ratios via DVA command, see Figure 4.52 (Minimaneuver Commands).

```
DVAwr PT.Gen.DL.FDiff.TrqRatio Abs 1 1
DVAwr PT.Gen.DL.CDiff.TrqRatio Abs 1 0.3
DVAwr PT.Gen.DL.RDiff.TrqRatio Abs 1 0
```

With the above command lines the torque is distributed 30% to the front axle (70% to the rear axle). The front differential is set to "1", this means that the torque is completely transferred to the left wheel. The rear differential is set to zero, this means that the torque is completely transferred to the right wheel, you can also see the tire force in the x-direction shown by the green force bar.

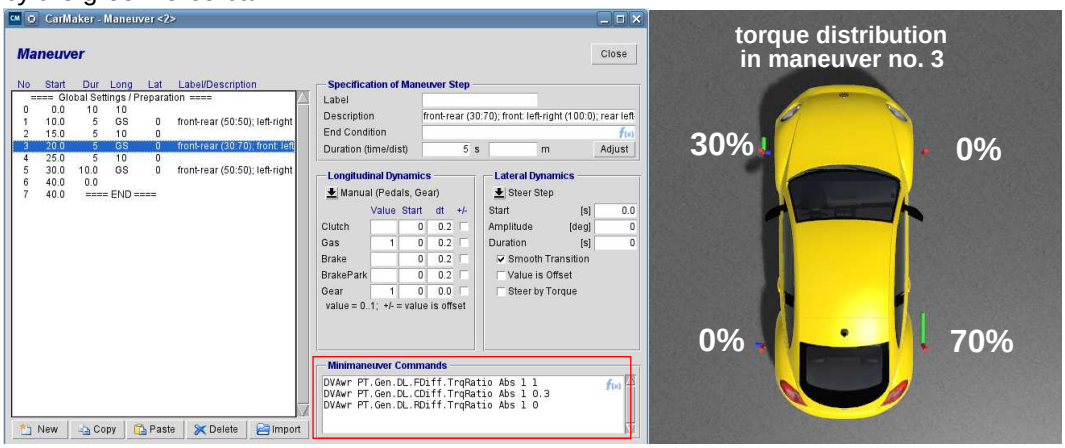


Figure 4.52: Maneuver for DVA controlled torque vectoring

Chapter 5
Vehicle Dynamics

5.1 QuickStartGuide

The TestRuns in this folder are the solutions of the exercises in Quick Start Guide section 5.1 'TestRun with Main Focus: Vehicle Dynamics' on page 25.

Step1_Scenario

This example shows how your TestRun should look like after following the instructions of Quick Start Guide section 5.1.1 'Building a Road Network Using the Scenario Editor'.

Step2_Maneuver

This example shows how your TestRun should look like after following the instructions of Quick Start Guide section 5.1.2 'Defining the Maneuver'

Step3_Vehicle

This example shows how your TestRun should look like after following the instructions of Quick Start Guide section 5.1.3 'Selecting a Vehicle and Simulating'.

Final_VehicleDynamics

This example shows how your TestRun should look like after following the instructions of Quick Start Guide section 5.1.4 'Saving Your TestRun'.

5.2 Longitudinal Dynamics - Braking

Braking

The braking test is one of the standard test procedures for vehicles. The test is performed by default on a dry road surface with homogenous friction coefficient. The vehicle test speeds are in the range of 80 - 100 km/h.

The features of the TestRun are

- acceleration from standstill
- the manual application of the brake pedal in the "***Longitudinal Dynamics***" *user interface of the maneuver control.*

The road is created with two segments, the wide of the second segment is adapted by the option "*Override selected Segment Attributes*" in the *Main GUI> Parameters > Road > Segments* interface. Thus the road in the braking area broadens in lateral direction.

The braking distance is measured by the Quantity **DM.ManDist**, which contains the driven distance in the actual maneuver. It will provide the distance since the braking maneuver (Maneuver No. 1) is started. An additional virtual braking point is defined by *Road Marking* and *Pylons, see* Figure 5.1. The relative vehicle position to the braking point on the road has no influence on the braking distance measurement. The **Car.Road.sRoad** contains the vehicle road coordinate along the route. The reference point lies in the middle of the front axle.

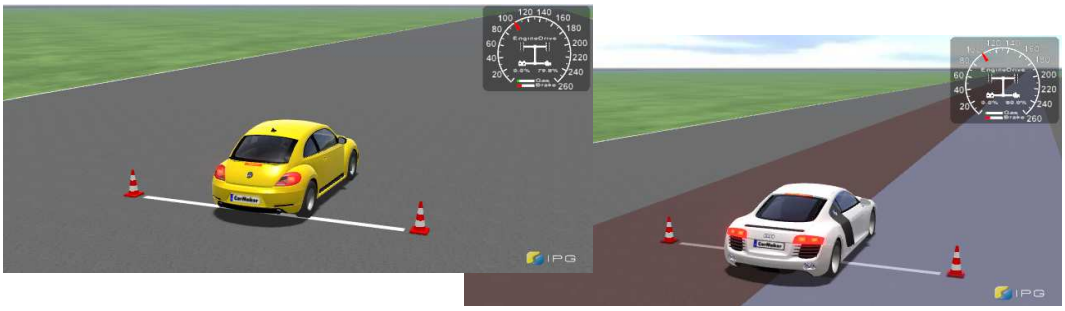


Figure 5.1: Braking Test with defined friction coefficient and additional Speed and Powertrain information in IPGMovie

BrakingVariation.ts

The TestRun is made for braking tests on mue-low, mue-high and mue-split road conditions. The mue-split braking defines a difference of the friction coefficients on the left and right side. This makes it challenging for stability control systems and antilocking systems. Vehicles without stability control systems (ESC) will spin around during a hard braking maneuver.

BrakingMuSplit

The features of the TestRun are

- creation of different friction coefficients on the road and the
- preparation of the TestRun (NamedValues) for test automation
- measuring the braking distance by Realtime Expressions

The different friction coefficients are defined in the Main GUI> Parameters > Road > Segments. The friction coefficient of the left side of segment no. 1 is set to *$Mue_left=0.5* and the right side to *$Mue_right=0.* The $ describes a NamedValue which can be accessed via TestManager or ScriptControl. The other NamedValues are the *$Speed* in the acceleration maneuver and the brake pedal value *$Brake* in the maneuver "Braking to standstill".

The brake distance is measured by the Minimaneuver command of the braking maneuver

```
Eval (Car.v<=0.001) ? Log ("Braking Distance: %.2f m",DM.ManDist)
```

If the velocity of the car is less than 0.001 m/s the braking distance is written to the session log by the command `Log ("Braking Distance: %.2f m",DM.ManDist)`. The quantity DM.ManDist provides the driven distance of the actual maneuver.

The usage of the NamedValues are shown in the TestSeries **BrakingVariaton.ts.** The features of the TestSeries are

- parameter variation of the road friction coefficient via **NamedValues** defined in the *Road > Segments > Override selected Segment Attributes*
- the variation of the driving speed and the brake pedal value.

The TestManager shows the variation of the parameters: *Speed, Brake, Mue_left* and *Mue_right* (see (2)). These parameters are defined in the TestRun **BrakingMuSplit** as NamedValues. The selection of the test car and the tire is shown in first section (1.) in Figure 5.2.

The TestRun variation can be added by the "Add" button. The Named Values can be defined as shown in section (4.), see Figure 5.2. The type NValue is selected, the name of the NamedValue and the value is defined in the following input fields.

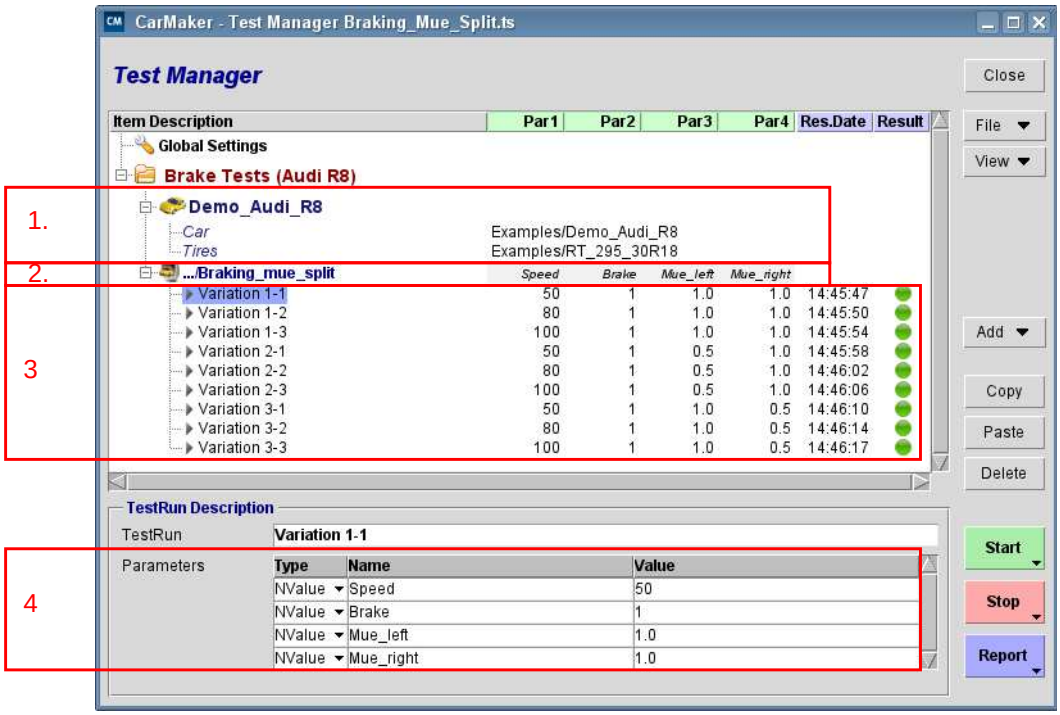


Figure 5.2: Variation of the BrakingMuSplit TestRun

The TestRun ABS_BrakingMuSuddenChange is also included in the TestSeries Braking-Variation.ts. The TestSeries shows a variation of the speed and the friction coefficient of the low mue area. Further information about the TestRun are prepared in the section ABS_BrakingMuSuddenChange.

BrakingFrictionPatchwork

to be documented

The TestRun shows a braking maneuver with fast changing friction coefficients of the road. The test velocity is by default 100 km/h. The friction coefficients differ on the left and right side. The test is made for testing ESC ECU on a road with fast changing friction coefficients.

The CarMaker features in this TestRun are

- changing the friction coefficients on the road surface
- initiation of a braking maneuver

First the driver accelerates the car to the "Cruising Speed" of 100 km/h (see, *Main GUI> Parameters > Driver > Standard Parameters),* because there is no speed defined in the "Accelerating" maneuver, see *Main GUI> Parameters > Maneuver.* After reaching the defined duration (in time or dist) the initiation of the manual braking maneuver "Braking" starts.

The different friction coefficients on the road are defined in the *Main GUI> Parameters > Road > Segments.* The braking area of the road is built of straight road segments with a length of 5 m. The friction coefficients of the segments are overwritten by the feature "*Override selected Segment Attributes*". The consecutive changing of friction coefficients results in the shown friction patchwork on the road.

CornerBraking

The TestRun shows a braking maneuver while driving through a corner.

The CarMaker features of the TestRun are

- braking maneuver
- logging and calculating quantities using "Realtime Expressions"

The driver accelerates the car to the defined "Cruising Speed" of 100 km/h (see, *Main GUI> Parameters > Driver > Standard Parameters),* because there is no speed defined in the "Accelerating" maneuver, see *Main GUI> Parameters > Maneuver* while driving on the first section of the corner.

In the "Minimaneuver Commands" of the first maneuver some logging outputs and new quantities are generated.

```
1:  Log "  vFzg: $vFzg=100 km/h"
2:  Log "   mue: $mue=0.6"
3:  Log "Radius: $r=300 m"
```

Line 1. to 3. printing some information to the *Session Log.* (see Ctrl-l)

```
4:  Eval Qu::max_YawRate=0
5:  Eval Qu::Brakedistance=0
6:  Eval Qu::YawRate_stat=0
```

Line 4. to 6. generates new quantities which are accessible in the IPGControl application, Main GUI> File > IPGControl.

```
7:  Eval Qu::YawRate_stat=abs(Car.YawRate)
```

In line 7, the quantity YawRate_stat is in each cycle overwritten by the absolute value of the Car.YawRate quantity.

The first maneuver ends if the following "*End Condition"* is complied with.

```
Car.Road.sRoad > 525
```

The braking maneuver is defined by a manual control of the longitudinal dynamics. The value of the clutch and the brake pedal are set to 1 (pressed).

In the minimaneuver commands the following lines are written to save the maximum value of the yaw rate and to measure the braking distance.

```
1:  Eval Qu::max_YawRate=max(abs(Car.YawRate),abs(max_YawRate))
2:  Eval Qu::Brakedistance=max(DM.ManDist,Brakedistance)
```

With the function max(a,b) the maximum of a or b is written to the defined quantity "max_YawRate" and "Brakedistance".

CornerBrakingMuSplit

This TestRun shows a variation of the TestRun CornerBraking. The difference is in

- the friction coefficient on the road in the braking area,
- the "End condition" of the acceleration maneuver and
- the road is switched to a right turn.

The following description is only about the changes according to the CornerBraking TestRun. The friction coefficient of segment no. 1 is overwritten by the a friction stripe, see *Main GUI> Parameters > Road > Segments.* In the user interface of the road segments you can also see the segment kind "Turn Right".

The first maneuver ends if the following "*End Condition"* is complied with.

```
Car.muRoadFR < 1
```

So if the road friction coefficient at the tire "FR = Front Right" sinks below 1 the braking maneuver is initiated.

StraightUpDown

to be documented

5.3 Lateral Dynamics - Handling

SteerImpulse

Vehicles are meant to stabilize themselves following a sudden steering impulse. To evaluate this vehicle characteristic, a series of steering impulses are performed in this TestRun. Following these impulses the steering wheel is released at maximum steering angle.

The magnitude of the steering impulses ranges from 1 to 5 m/s^2 in steps of 0,5 m/s^2. Their duration is set to 0,2s in accordance with ISO 17288-2. After each impulse the steering wheel is released for 5s. During this time the vehicle should return to a steady state.

To display relevant quantities the predefined diagram "Steering" can be selected in IPGControl (*Right-Click > Display Diagram > Steering*), see Figure 5.3.

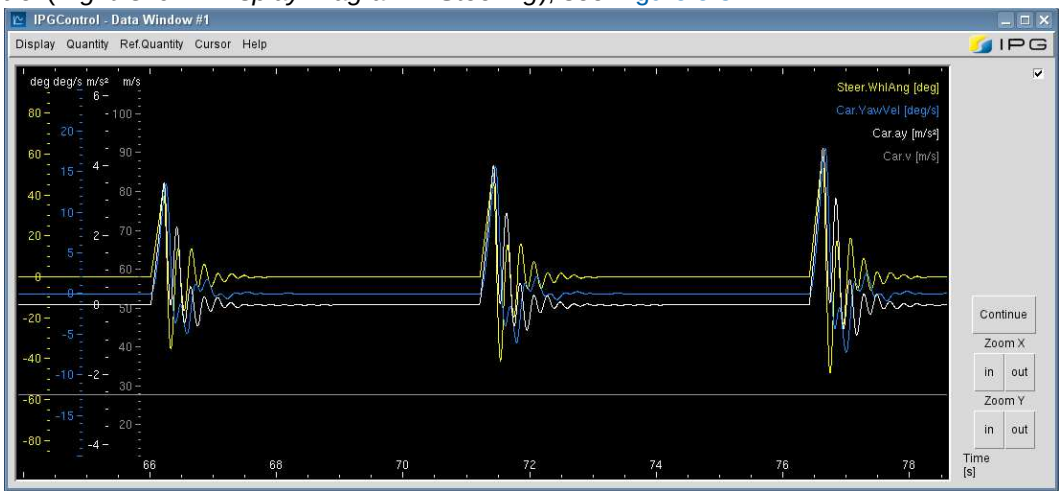


Figure 5.3: IPGControl (Steering Angle, Yaw-Rate, Lateral acceleration, Velocity)

In this example, the DemoCar is used. If other vehicles are used the amplitudes of the steering impulses have to be adapted. The amplitudes to be used have to be determined in a Pre-Test. To obtain the respective amplitudes, record the lateral acceleration reached by the vehicle using a specific steering angle together with the respective steering angle. Then, the amplitudes of interest can be calculated using Interpolation.

SteerStep

The steer step test according to ISO 7401 shows a test scenario for testing the lateral dynamic behavior of the vehicle.

The features of this TestRun are

- the implementation of a steer step maneuver with predefined lateral acceleration by
- using Realtime Expressions and dynamic maneuver "End Conditions"

The TestRun starts with an acceleration maneuver to the velocity of 80 km/h. To save time the following "End Condition" is used. The acceleration maneuver ends, if the car reaches the velocity of 80 km/h (Car.v [m/s])

```
Car.v>=80/3.6
```

In maneuver no. 1 a slowly increasing steer step is performed with a steering angle speed of 1°/s. If the lateral acceleration of the car is greater or equal 4 m/s^2 the current steering angle is stored by the following command, see 1. in Figure 5.4.

```
Eval first (Car.ay >=4) ? Qu::SteerAngle=Steer.WhlAng
```

The prefix `Eval` indicates a Realtime Expression. The steering angle is written to the created quantity Qu::SteerAngle. Before performing the actual steer step maneuver the vehicle should get to a steady-state condition while driving straight with the defined speed.

The steer step is performed with the lateral dynamic maneuver "Steer Step", see *Main GUI> Parameters > Maneuver.* The steering angle velocity is defined to 500 °/s. After reaching the pre calculated steering angle Qu::SteerAngle (see End Condition 2. in Figure 5.4) the steering angle is held by the maneuver no. 4 with the duration of 10 s.

To avoid any input in longitudinal direction the pedal positions are "frozen" in the actual position (before the steer step) by activating the "value is offset" option, see 3. in Figure 5.4

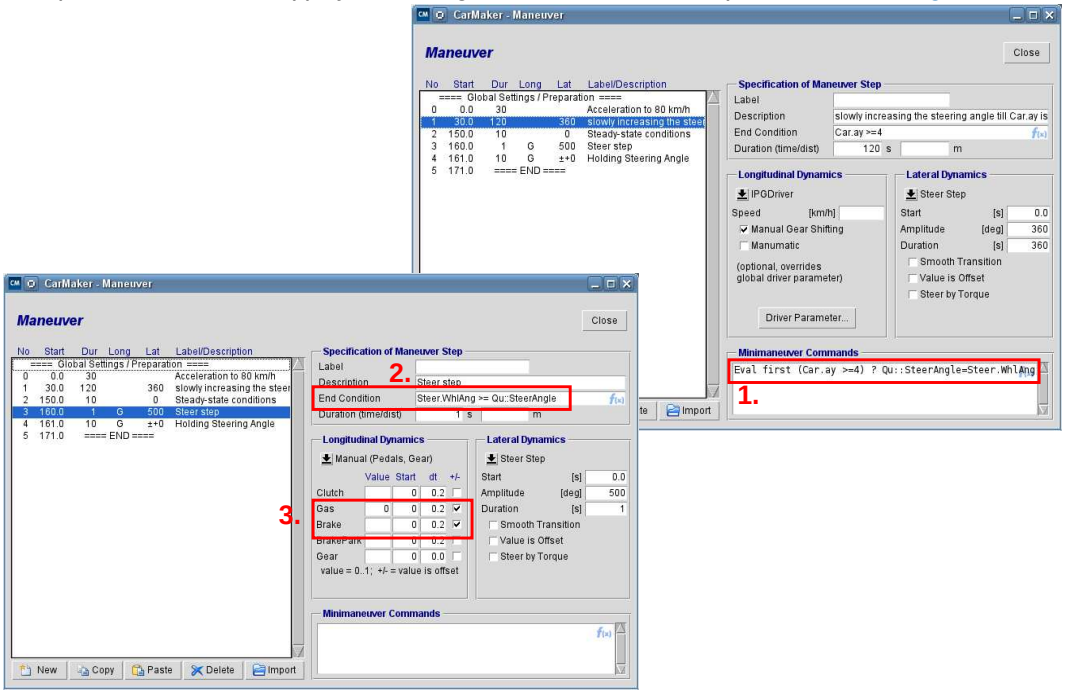


Figure 5.4: Steer Step maneuver

SinusSteering.ts

There are several tests to evaluate the dynamic characteristics of a vehicle. One test to evaluate the steering behavior is based on a sinusoidal steering input and is presented below.

This Test Series is based on these TestRuns:

SinusSteering_preTest

to be documented

SinusSteering

The TestRun "SinusSteering_preTest" consists of three maneuvers. First, the vehicle is accelerated to an initial velocity of 100 km/h. Second, a steady-state maneuver is executed for 10s. After the vehicle has reached a steady-state a defined steering maneuver is performed to evaluate the steering angle necessary for different lateral accelerations (maneuver No.2). This maneuver end when the maximal lateral acceleration is reached.

The determined steering angles are saved to user-defined CarMaker Quantities using the following Minimaneuver Commands:

The User Quantities are declared and initialized:

```
Eval first() ? Qu::SteeringAngle_2 = 0
Eval first() ? Qu::SteeringAngle_4 = 0
Eval first() ? Qu::SteeringAngle_6 = 0
Eval first() ? Qu::SteeringAngle_max = 0
Eval first() ? Qu::Max_Car_ay = 0
```

The following command is evaluated in each simulation cycle. If the current lateral acceleration is larger than the previous maximum value the maximum value is replaced.

```
Eval Car.ay >= Max_Car_ay ? Max_Car_ay = Car.ay
```

Next, the steering angles necessary to obtain the respective lateral accelerations are recorded.

```
Eval first (Car.ay > 2) ? SteeringAngle_2 = deg(Steer.WhlAng)
Eval first (Car.ay > 4) ? SteeringAngle_4 = deg(Steer.WhlAng)
Eval first (Car.ay > 6) ? SteeringAngle_6 = deg(Steer.WhlAng)
```

The last line records the steering angle needed to reach the maximal lateral acceleration.

```
Eval first (Car.ay < Max_Car_ay) ? SteeringAngle_max = deg(Steer.WhlAng)
```

In the TestRun "SinusSteering" several sinusoidal steering maneuvers are performed. Some parameters of this TestRun are replaced with NamedValues so that the intended variations can be generated in the TestManager, see Figure 5.5.

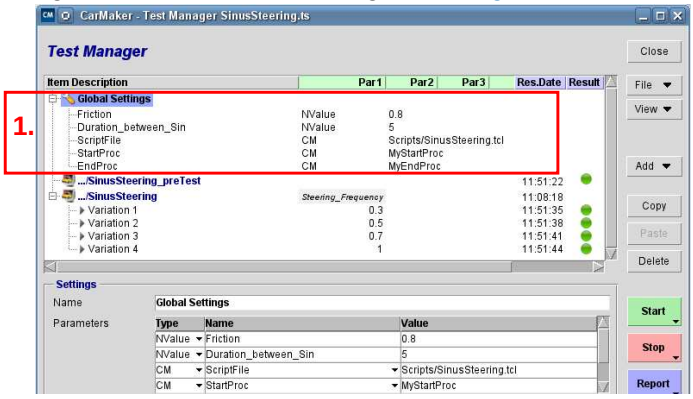


Figure 5.5: SinusSteering TestManager

In the TestManager NamedValues are defined. In the "Global Settings" section the friction coefficient of the track ("Friction") and the time interval between the sinusoidal steering maneuvers ("Duration_between_Sin") are set, see 1. in Figure 5.5. Additionally, the script file "SinusSteering.tcl" as well as the start procedure and end procedure are selected.

After the execution of the TestRun "SinusSteering_preTest", the procedure "Set_Sinus_Amplitude" is called. The following statement adds a quantity to the Scratchpad.

```
QuantAudit add SteeringAngle_2 -time 0-end
```

After this, the NamedValue "Amplitude_1" is overwritten with the maximal value of the variable "SteeringAngle_2".

```
NamedValue set Amplitude_1 [Scratchpad get QuantAudit-SteeringAngle_2 0 max]
```

At the beginning of the TestRun "SteeringSinus" the procedure "Set_Sinus_Duration" is called. It sets the NamedValue "Steering_Duration" in accordance with the previously defined Frequency.

```
NamedValue set Steering_Duration [expr 1/$TS::Steering_Frequency]
```

SteeringReleaseVariation.ts

The "Steering-release open-loop test method", according to ISO 17288-1, is a test case for the evaluation of the free-steer behavior of vehicles.

The steering behavior of vehicles should be stable in all cases. In this TestRun, the vehicle is driven on a steady-state circular-course with a constant lateral acceleration. Starting from this steady-state the steering wheel is released to evaluate the vehicle's reaction. The vehicle should return to a steady-state.

The TestRun consists of these maneuvers: First, the vehicle is accelerated to a velocity of 100 km/h. Second, steering is applied until the vehicle reaches a specific lateral acceleration. This lateral acceleration is varied starting with a value of 1 m/s^2 in steps of 1 m/s^2 until the maximal lateral acceleration is reached. When the currently specified lateral acceleration is reached the vehicle is driven with a constant steering angle for a duration of 10s to establish a steady-state. In the final maneuver, the steering wheel is released.

SteeringRelease

This TestRun demonstrates the following CarMaker features:

- Variation of parameters using the TestManager
- Using RealtimeExpressions to define NamedValues
- Using the Steer-by-Torque steering system *Main GUI > Parameters > Car > Steering > Steering Model > Dynamic Steer Ratio (GenTorque)*

In the first maneuver "Acceleration to 100 km/h" the vehicle is accelerated until the starting velocity of 100 km/h is reached. Next, steering is applied until the following condition is reached:

```
Car.ay >= $(LatAccel=5)
```

Here, $LatAcceleration is a NamedValue that can be varied within the TestManager. Using the bracket-notation a default value can be set. In maneuver number 2 the steering angle is fixed using a "SteerStep" with an amplitude offset of 0 degrees. In maneuver number 3 the steering wheel is released by usage of a Steer by Torque "Steer Step" maneuver with a magnitude of 0 Nm.

The vehicles behavior following the release of the steering wheel can be analyzed using IPGControl, see Figure 5.6.

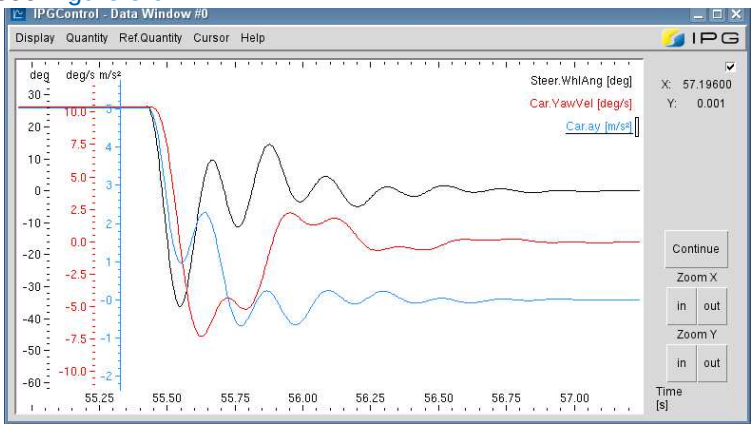


Figure 5.6: Vehicle response after steering release (Car.v = 100 kph, Car.ay=5m/s^2)

This TestRun is embedded in the TestSeries "SteeringReleaseVariation.ts". The TestSeries varies the target lateral acceleration within a range of 1 m/s^2 up to the defined maximal lateral acceleration.

LaneChange.ts

The double-lane-change maneuver is a closed-loop test maneuver. The TestSeries presents different "double lane change maneuver tests" which are defined according to the standards of the respective test organizations and standardizations.

The "LaneChange" test is used for evaluating the vehicles safety and handling ability. The primary function is to test the tilt stability of the vehicle. The course is defined by an entry-, a shifted- and an exit-pylon-alley. The width of the alley sections depends on the vehicle width (without side mirrors) or to fixed standards.

The TestSeries includes different test procedures:

LaneChange_ISO

LaneChange_VDA

LaneChange_AMS

LaneChange_ADAC

The *Lane_Change_ISO* (ISO 3888-1) maneuver and the *VDA Lane Change Maneuver* shows defined test-cases which are developed particular for testing the vehicle safety and stability characteristics. These two maneuvers differ from each other by the width of the pylon alleys and the lateral offset of the center passage. That implies a difference in the entrance speed. For the ISO Lane Change the entrance speed is much higher than for the VDA test.

The Series includes also the *AMS lane change test* of the "*Auto motor und sport (AMS)*" magazine, which has defined a own test maneuver according to the ISO and VDA Lane Change tests. The test is defined with a fixed pylon alley width.

The fourth test describes the avoidance test of the ADAC (Allgemeiner Deutscher Automobil-Club e.V.) (General German Automobile Club). The test vary widely from the other three tests. It simulates the collision avoidance test. An entrance speed of 90km/h (55.92 mph) is defined for each test **[11]**.

The TestSeries example includes the following CarMaker functions

- parameter variation and modification by TestManager,
- usage of a script file for preprocessing functions (calculation of pylon alley width) according to the actual TestRun
- usage of NamedValues for online modification of the TestRun

The vehicles width is set as *Testspace Variable* "Vehicle_Width" in the Global Settings (see Figure 5.7 (1)) of the TestManager. The *KeyValue* (KValue) "TestRun:Vehicle" defines the type of the vehicle. The path to the script file is specified by the CarMaker variable "ScriptFile", also the name of the starting procedure "StartProc" is defined in the Global Settings.

The TestSeries includes a parameter variation of the vehicles entrance speed, see Figure 5.7 (3). The NamedValue "Speed" is defined in the *Main GUI> Parameters > Driver > Standard Parameters > "Cruising Speed".* The "Speed" parameter is changed by the variation in the TestManager.

The single TestRun variation is rated by the criteria "Pylon Detection", see Figure 5.7 (2). The ScriptControl Command **`Scratchpad list PylonHit`** is used for the evaluation, if a pylon was hit by the car. If the list is empty, no pylon was hit and the TestRun is marked with a green (good) result symbol in the TestManager interface. The "Scratchpad list PylonHit" delivers all pylon-hits. For further information see*'Programmer's Guide chapter Managing the Scratchpad'.*

The special parameters **"PylonShiftFdCoef"** and **"PylonShiftBkCoef"** for lane change through pylon sections can modify the course of the driver. The parameters are located in *Main GUI> Parameters > Driver > Misc./ Additional Parameters* additional these are linked to the NamedValues in the Global Settings of the TestManager. For good results, the parameters should stay between 0 and 0.3. The first parameter shifts the course forwards the second parameter shifts the course backwards. Further information are provided in 'IPGDriver Reference Manual chapter Modification of the Course'.

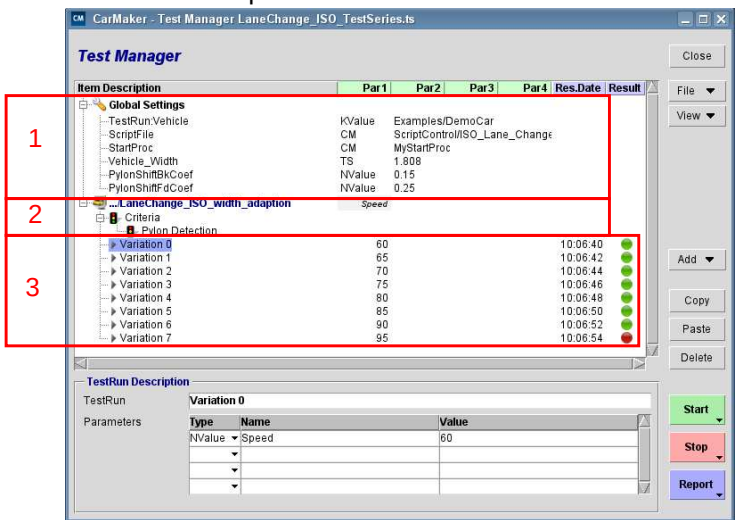


Figure 5.7: ISO Lane Change TestSeries with adaptive pylon alley width

Slalom18m

to be documented

Slalom18m_AMS

to be documented

Slalom36m

These tests are used for the objective and subjective evaluation of the vehicle dynamic. As a objective characteristic of the vehicle's performance the time needed to complete the slalom parcour as well as the mean velocity are measured. Thus, the performance of different vehicles can be compared objectively. The tests "Slalom_18m" and "Slalom_18m_AMS" differ with respect to the way the vehicle enters the slalom course.

The TestRun relies on the following CarMaker features:

- the pylon course definition
- RealTimeExpressions for calculation and measurements

In CarMaker pylons are generally used in pairs and build something similar to a door that is integrated into the calculation of the course the driver takes. To position the pylons on the middle line a lateral offset of + 0.25 m is used. Positive offset values move the pylons to the left.

Figure 5.8: 180m Slalom course (AMS)

To pass the slalom parcour as fast as possible, the lateral acceleration parameters of the driver have to be adapted. The values chose here seem not to be realistic (see Figure 5.9), but they are useful to make the driver drive the vehicle at its dynamic limits. Further improvements can be obtained by varying the Corner-Cutting-Coefficient so that the driver drives closer to the pylons.

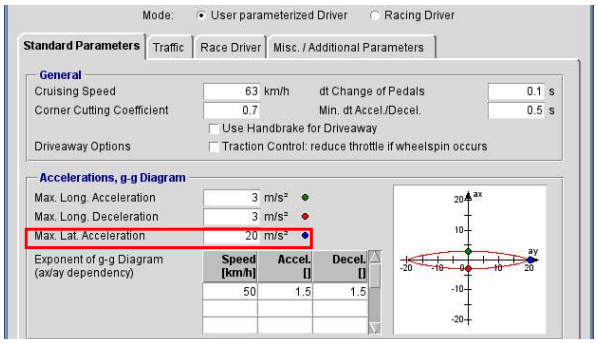


Figure 5.9: Driver parameter for Slalom 18m (AMS)

Racetrack_Hockenheim

The TestRun shows the small race track of the grand prix circuit "Hockenheim". The 2598 m long race track will be driven by the "Racing Driver" in the Demo_McLaren_F1 car. The driver is set to the mode "Racing Driver", see Main GUI> Parameters > Driver. The knowledge of the racing driver is shown in Figure 5.10. For more information about the Race Driver see IPGDriver User Manual chapter Race Driver.

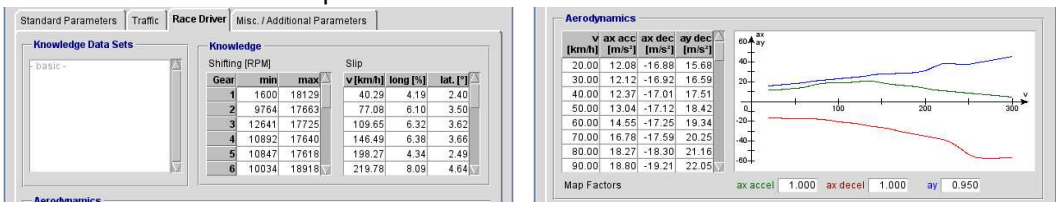


Figure 5.10: Racing Drivers knowledge about wheel slip, shifting and aerodynamics

With the racing driver the lap time is printed in the Session Log, see (Ctrl-l / Main GUI > Simulation > Session Log).

Racetrack_Nurburgring

This TestRun demonstrates a simulation of the well-known racing track "Nordschleife". The circuit is included as a digitalized road, see *Main GUI> Parameters > Road > General Settings > Digitized Road*.

The TestRun includes the following CarMaker features:

- importing a digitized road
- IPGDriver mode "Racing Driver

The vehicle is controlled by the IPGDriver in "Racing Driver" mode. Due to its knowledge about the dynamic limits of the vehicle (see *Main GUI > Parameters > Driver > Racing Driver)* the vehicle can be driven at its dynamic limits throughout the complete track.

The knowledge about the vehicle's dynamic limits are learned by the Racing Driver using Driver Adaption, see *Main GUI > Simulation > Driver Adaption*. Further information can be found in the IPGDriver user manual.

The track is prepared with signs, curbs, guard rails and trees which are defined as road attributes (Bumps/Markers) on the digitized road. How to load digitized roads into CarMaker is shown in *Users Guide chapter 4.4.4 Digitized Roads.*

Figure 5.11: Digitized Road 'Nordschleife' with terrain

The terrain is included by the option "Terrain", see Main GUI> Parameters > Road > IPG-Movie Interface > Terrain.

LapTimeOptimization

The TestRun demonstrates the approach of the internal optimization of your laptime. For demonstration purposes the small race track of the grand prix circuit "Hockenheim", a 2598 m long race track, will be used.

Read more about lap time optimization in the "**IPGDriver User Manual** in the chapter *Use Case: Lap Optimization*" and CarMaker User Manual chapter *Learning Procedure: Driver Adaption*"

Steps for the Laptime Optimization:

- Start with the *Driver Adaption* to get basic knowledge of the handling and the vehicle limits.
- Optimize the Laptime by increasing the default speed-dependent exponents of the g-g Diagram

SteadyStateVariation.ts

to be documented

SteadyStateCircular42m

to be documented

SteadyStateCircular100m

The quasi-steady-state circular test analyzes the vehicle's dynamic behavior depended on lateral acceleration by providing information about it's self-steering effect (understeer/ oversteer). The target behavior is described by a linear progression of the steering angle depending on the lateral acceleration. Additionally, there should be a progressive gradient close to the vehicle's dynamic limits so that these limits are pointed out to the driver.

The test can be performed either in a stepwiese mode or with a constant increase in velocity. In this TestRun the second option is applied. It demonstrates the following CarMaker features:

- maneuver End Condition for the detection of to high deviation from the circle path
- **DVA command** to overwrite the gear selection of the IPGDriver

To avoid shifting in the test drive, the driver moves of in the first gear, after a few seconds the drivers gear selection will be overwritten by DVA command.

```
[Time>3] DVAwr VC.GearNo Abs -1 3
```

The maneuver is finished after the quantity **Car.Road.Path.DevDist,** which shows the deviation of the circular path, is greater than 0.5 m (absolute).

```
abs(Car.Road.Path.DevDist) > 0.5
```

The maximum of the longitudinal acceleration is set in the driver accelerations in *Main GUI> Parameter > Driver > Max.Long.Acceleration* to 0.2 m/s^2.

Eight

The TestRun shows a course of the road shaped like a eight. The driver follows the course by first driving into the left turn and afterwards into the right curve. The end of the track is designed to end in the left curve. So the driver will follow the designed eight course over and over again as it is always in the case of a closed circuit.

The road segments are designed in the segments interface, see *Main GUI> Parameters > Road > Segments.*

SpeedOval

The TestRun shows a speed oval with banked 180 degree curves. The road is created by segments, see *Main GUI> Parameters > Road > Segments*. The banked turns are created by a negative slope see Figure 5.12.

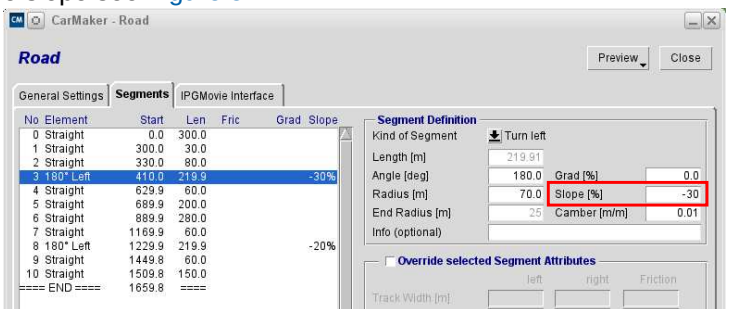


Figure 5.12: Speed oval with banked turns created with road segments

StreetsOfSanFrancisco

The TestRun shows a challenging road course with high gradients and serpentine course segments.

The Features of the TestRun are

- creation of serpentine road parts
- uphill and downhill segments
- flying vehicle part at the end of the TestRun

In the first maneuver the vehicle is driven by the IPGDriver (closed-loop maneuver). After 1885 m (before flying part) the throttle is set manual to the actual gas pedal value for the duration of the second maneuver, see duration 100 m in the maneuver interface Main GUI> Parameters > Maneuver. After 100 m the vehicle control is set again to the IPGDriver (closed-loop)

Figure 5.13 shows the flying part of the TestRun. The red line shows the position where the first maneuver ends.

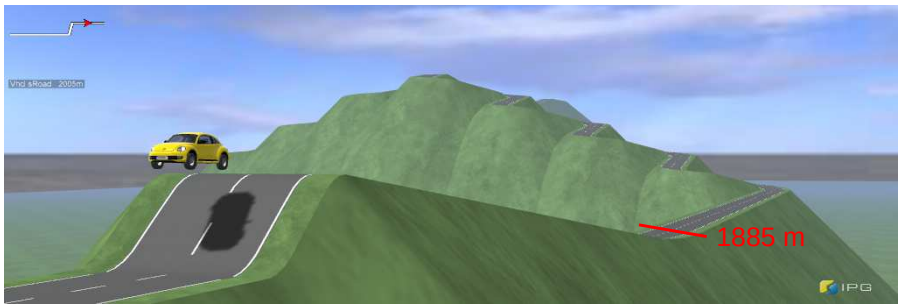


Figure 5.13: Flying part of the TestRun

5.4 Stability Control

ABS_BrakingMuSuddenChange

The TestRun shows a braking maneuver which starts on the road area with low friction coefficient. During the braking maneuver the vehicle drives out of the low mue area onto the high friction area. The TestRun is developed to produce this sudden change of the road friction coefficient for testing ABS/ESP controller.

The friction coefficient of the road can be changed in the road user interface, see *Main GUI> Parameters > Road > Segments* and Figure 5.14*.* The first part overrides the track width and the second part, see 2. Figure 5.14, creates the low mue area.

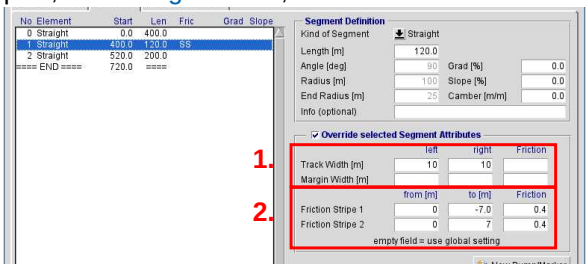


Figure 5.14: Road segment user interface

ASR_AccelerateMuSuddenChange

The TestRun shows an acceleration maneuver starting on a low mue surface. While accelerating the vehicle drives off the low mue area onto the road with normal friction coefficient (mue=1).

The TestRun is designed for testing the behavior of ASR controllers in the condition of sudden changing friction coefficient of the road.

By testing the traction control system it is important to switch off the drivers internal "Traction Control" (see Figure 5.15) in the drivers user interface, see Main GUI> Parameters > Driver.

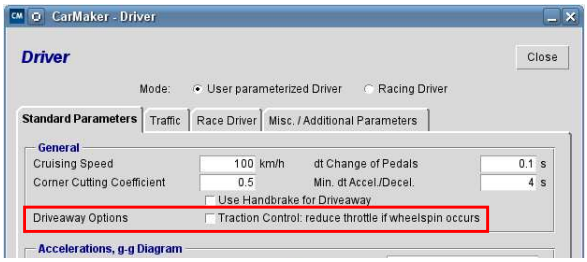


Figure 5.15: IPGDriver's internal traction control

ASR_AccelerateMuSplit

The TestRun shows an acceleration maneuver starting on a mue split surface. On the right side the friction coefficient is set to 0.4. On the left side the coefficient is 0.9. While accelerating the vehicle drives off the mue split area onto the road with normal friction coefficient (mue=1).

The TestRun is designed for testing the behavior of ASR controllers in the condition of different road surfaces.

By testing the traction control system it is important to switch off the drivers internal "Traction Control" (see Figure 5.15, TestRun **ASR_AccelerateMuSuddenChang**) in the drivers user interface, see Main GUI> Parameters > Driver.

ESC_ParkingGarage

The TestRun is designed for testing ESC systems. Sometimes the combination of a steep driveway and additional sharp turns (see Figure 5.16) causes the activation of the electronic stability control (ESC).

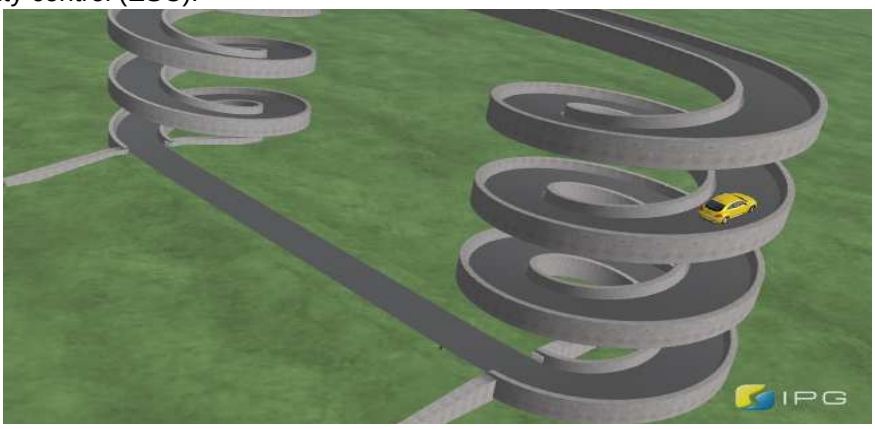


Figure 5.16: Parking ramp for ESC testing

The track is created by road segments, see *Main GUI> Parameters > Road > Segments.* The DemoCar is selected for demonstration of this TestRun. For realistic tests a ESC system in hardware (HIL) or software (User extension C-code-model, Simulink, FMU-PlugIn) should be integrated.

ESC_SineWithDwell.ts

to be documented

ESC_SineWithDwell

to be documented

ESC_SineWithDwell_PreTest

The test SineWithDwell is defined in the guideline "FMVSS No. 126 Electronic Stability Control Systems" **[9]** by the National Highway Traffic Safety Administration (NHTSA). It is used for homologation purposes for ESC systems. The test provokes oversteering behavior in a vehicle. Thus, it causes the Electronic Stability Control to intervene.

The test is implemented using these CarMaker features:

- Variation of NamedValues and generation of diagrams in TestManager
- Declaring new variables (quantities) in the CarMaker Data Dictionary
- Using RealtimeExpressions to perform a maneuver
- Using a ScriptFile for pre- and post-processing purposes

The TestSeries starts with the TestRun "SineWithDwell_PreTest" (slowly increasing steer). In this pre-test the steering angle necessary to reach a lateral acceleration of 0.3 g is estimated.

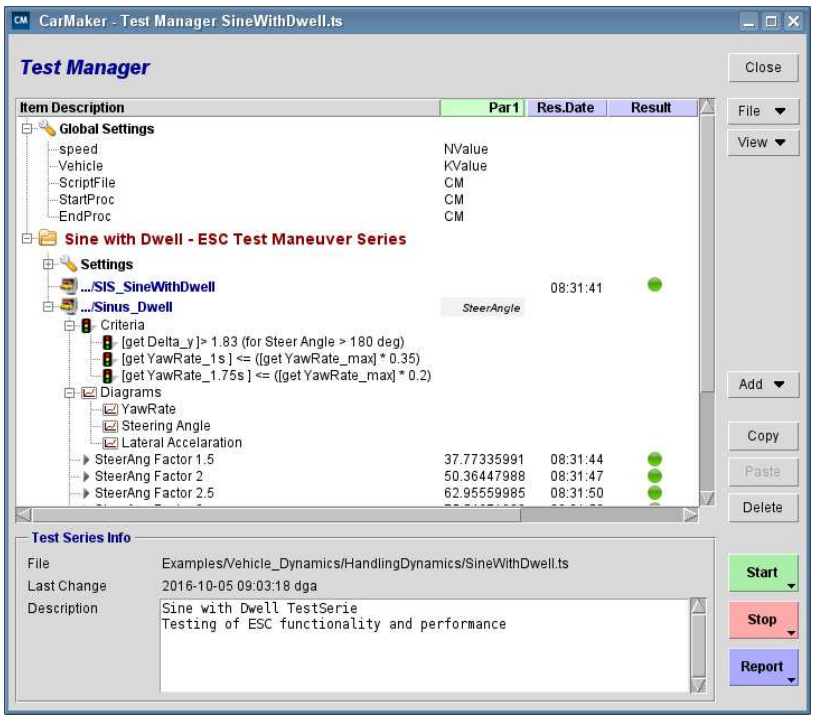


Figure 5.17: Sine with Dwell TestSeries

The TestSeries uses the ScriptFile "SineWithDwell.tcl" (Scripts/SineWithDwell.tcl). This ScriptFile has two core features. Before the pretest "SineWithDwell_PreTest" is performed, it sets the NamedValue controlling the lateral acceleration (ay) to the predefined value of "Acceleration_3g" (0.3 g / 2.943 m/s^2). After the pre-test is done, it evaluates the steering angle needed in the pre-test and generates the variations of the actual SineWithDwell TestRun based on this information. This is done in the procedure "Evaluate_SIS". Additionally, this procedure deletes all previously defined variations.

In the variations of the "SineWithDwell" TestRun the steering angle is varied based on the steering angle determined in the pre-test (steering angle needed to obtain a lateral acceleration of 0.3g). In the first variation this steering angle multiplied by a factor of 1.5 is applied. In the consecutive variations, the multiplicational factor is increased in steps of 0.5 until it reaches a value of 6.5 in the last variation. The final steering angle should exceed 270 deg but must not exceed 300 deg. If it is smaller than 270 deg the steering angle is set to 270 deg in the last variation (see "proc EvaluateTest_SIS" in the ScriptFile).

The TestRun "SineWithDwell" begins with an acceleration maneuver to a velocity of 80 kph. In the next maneuver a steady-state should be reached.

Next, sinusoidal steering with a frequency of 0.7 Hz and the pre-defined steering angle is applied. At three-quarters of the sinus's period the steering angle is fixed for 500 ms. Finally, the sinusoidal steering is resumed until the full period duration is reached.

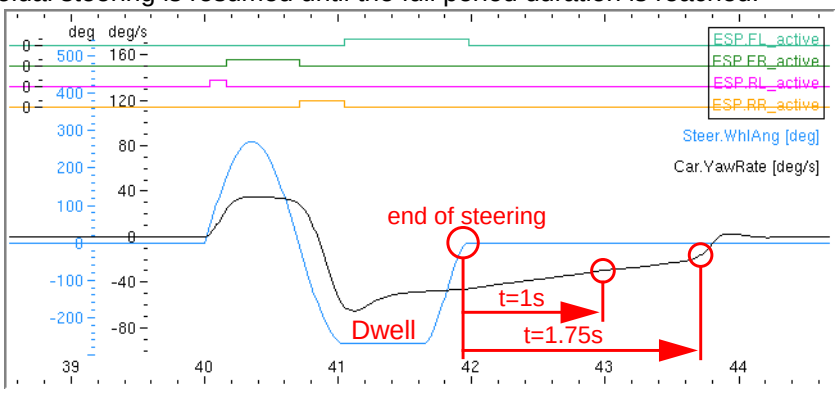


Figure 5.18: Steering Wheel Angle and Yaw Rate in the sine with dwell maneuver (left-right)

The vehicle used in this test should be equipped with an ESC ECU. In this TestRun the Simulink model "HydBrakeCU_ESP" is used (further information can be found in the chapter *HydBrakeCU_ESP* of the Programmer's Guide).
Figure 5.6 visualizes a Sine with Dwell maneuver (left-right) including ESC interventions to all four wheels. Due to the sudden steering maneuver to the left understeering is provoked. Thus, braking is applied to the rear wheel on the inside of the corner (RL). After the next steering maneuver, the vehicle is showing a tendency to oversteer. Thus, braking is applied to the front wheel on the outside of the corner. In the second half of the sinus's period steering to the opposite side of the road is applied and the ESC interventions are repeated in a curve to the right.

There are three criteria that have to be fulfilled by the ESC system according to **[9]** so that the variation can be evaluated as "good":

- Responsiveness Criterion: The vehicle's lateral displacement (ty) should be above 1.83m (6 feet) 1.07 seconds after the steering maneuver has started (for vehicles weighting above 3500 kg -> 5 feet (1.22 m)). This criterion is meant to ensure that vehicles are responsive during a ESC intervention. It is applicable for maneuvers with a steering amplitude above 180 deg.
- 1st Lateral Stability Criterion: The vehicle's yaw rate shall not exceed 35 percent of the previous maximum value 1 s after the end of the steering maneuver.
- 2nd Lateral Stability Criterion: The yaw rate must be below 20 percent of its maximal value 1.75 s after the end of the steering maneuver.

If one of these criteria is not met in a TestRun the following lines of code skip to the end of the current group. Other options to use with the command TestMgr::SkipToEnd are <TestRun>, <TestSeries>, <Group> or <TestRunGroup>. Further information can be found in the chapter Testautomation (section ScriptFile) of the CarMaker User's Guide.

```
if {[TestMgr get Result] == "bad"} {
    TestMgr::SkipToEnd Group
}
```

ESC_TrailerOscillation

This TestRun demonstrates an OpenLoop driving maneuver with a trailer. After acceleration the assembly of car and trailer is exited to oscillate by a sinusoidal steering maneuver. In this example, the vehicle "DemoCar" that is not equipped with a stability control system is used. By the end of the TestRun the vehicle has left the track.

The TestRun demonstrates these CarMaker features:

- Adding a trailer
- Applying sinusoidal steering using a *Lateral Dynamics* maneuver
- Stepwise acceleration until the assembly's limit of stability is reached.

The trailer can be selected in the menu *Main GUI* > *Parameters* > *Trailer*. The position of the trailer hitch is defined in *Main GUI* > *Paramerters* > *Car* > *Bodies* > *Hitch.*

Informations about the Trailer model are presented in "CarMaker User's Guide chapter *Trailer Model* ".

In several countries there are reduced velocity limits for assemblies of car and trailer. This TestRun can be useful to determine the limiting velocity above that the trailer starts swinging up. Additionally, in this test the oscillation is encouraged by incorrect dimensioning of the drawbar.

ESC_TrailerOscillationVariation.ts

The test series shows a variation of speed for testing the trailer stability in the case of inappropriate steer angle.

5.5 Rollover

FishHook

This test case shows the use of dynamic maneuver end conditions in conjunction with the construction of the roll rate-based "Fishhook" maneuver.

The functionalities used within this TestRun are

- the creation of new "user accessible quantities" and
- the definition of dynamic "End Conditions" for maneuver steps.
- Saving the value of quantities and calculation in realtime expressions.

The TestRun example demonstrates the fishhook maneuver with roll rate-induced steering reversal, which is made for testing the rollover stability of SUVs. The TestRun is based on dynamic end conditions and describes how to set dynamic end conditions, calculate in realtime expressions and use calculated values for following maneuvers.

```
Eval Qu::SteerWhlAngLimit=0
```

The maneuver no. 0 starts with accelerating to the specified velocity of 55 kph. The quantity SteerWhlAngLimit is created. After accelerating to the testing speed maneuver no. 1 is performed to ensure a static initial situation.

```
abs(Car.ay)>=2.943
```

Maneuver no. 2 ends when a lateral acceleration of 0.3g (2,943 m/s^2) is reached.

```
Eval SteerWhlAngLimit=6.5*Steer.WhlAng
```

In the following maneuver the current steering angle is multiplied with a factor of 6.5 and saved to the pre-defined variable. After this, a straight driving maneuver is applied to reach a steady-state starting point for the next steps.

```
DM.Steer.Ang>=SteerWhlAngLimit
```

Now, the "Fishhook" maneuver is initiated. A steering velocity of 720 deg/s is applied until the previously calculated steering angle "`SteerWhlAngLimit`" is reached.

```
abs(Car.RollVel)<=0.02618
```

In the second part of the "Fishhook" maneuver, the steering angle is fixed until the roll velocity of the vehicle falls below 1.5 deg/s. After this, countersteering is initiated.

```
DM.Steer.Ang<=-SteerWhlAngLimit
```

Countersteering is applied with a steering velocity of 720 deg/s until the previously defined steering angle is reached. Then, the steering angle is fixed for a duration of 3s.

Finally, the steering wheel is moved back to its central position.

LiftOff_PreTest

The TestRun is designed as PreTest for finding the maximum speed for driving on a circle with a radius of 30 m.

The features of the TestRun are

- defining maneuver end condition
- setting the drivers acceleration limits

The driver performes a slowly increasing velocity test on a circular course. The driver increases the velocity with the defined acceleration limits, see in the driver user interface under Main GUI> Parameters > Driver.

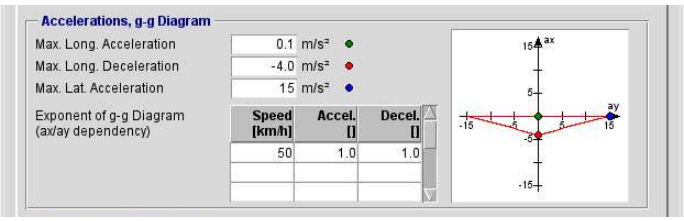


Figure 5.19: Acceleration limits of the driver

The maneuver end condition, see following line, defines the end of the maneuver if the absolute value of the lateral deviation from the desired static course (`Driver.Lat.dy`) is greater than 0.5 m.

```
abs(Driver.Lat.dy)>0.5
```

In the minimaneuver commands the maximum reached velocity is saved by the command lines

```
1:  Eval first() ? Qu::SteadyState.MaxSpeed=0
2:  Eval SteadyState.MaxSpeed=max(Car.v*3.6,SteadyState.MaxSpeed)
```

The first line defines the new quantity `SteadyState.MaxSpeed`. The second line saves the maximum driven velocity to the defined quantity.

LiftOffOversteer

The TestRun shows a steady state driving maneuver on a circular track. If the defined conditions are reached the throttle is lifted while the steering angle stays at the value before the lift-off.

The TestRun shows the features of

- jumping between maneuvers
- manual pedal control
- maneuver end conditions

A new quantity "Qu::StartCondNotReached" is created in the first maneuver.

```
Eval first () ? Qu::StartCondNotReached=0
```

The longitudinal dynamics are controlled by setting the brake pedal value to 1.

The maneuver no. 1 slowly increases the velocity to the defined value. After the second gear is selected the gear selection control is set by DVA.

If the second gear is selected, the first line will set the gear selection control to manual gear selection. The second line sets the actual target gear to the gear number 2.

```
[PT.GearBox.GearNo==2] DVAwr DM.SelectorCtrl Abs DM -1 2
[PT.GearBox.GearNo==2] DVAwr PT.Control.GB.GearNoTrg Abs DM -1 2
```

The maneuver control jumps with the function DMjmp "name" to the maneuver with the lable "beforeLO", if the centripetal acceleration (according ISO 8855:2011, 5.1.15) is below 0.01 and in the same time the speed is above 99% of the determined velocity $MaxSpeed (NamedValue) or the default value of 51.3 kph.

```
[Car.atHori<0.01&&Car.v>0.99/3.6*$MaxSpeed=51.3 ] DMjmp beforeLO
```

Premature end of the TestRun: If the condition above is not achieved the maneuver no. 1 will end after 60 s. In maneuver no. 2 the quantity *Qu::StartCondNotReached* is set to 1 and the DMjmp command will jump to the last maneuver with the lable "end".

With the maneuver no. 3 the speed is hold at a constant speed for 5 seconds before the throttle is lifted of with a manual maneuver in maneuver no.4. In addition the steering wheel angle is hold at the actual position., see 2. in Figure 5.20.

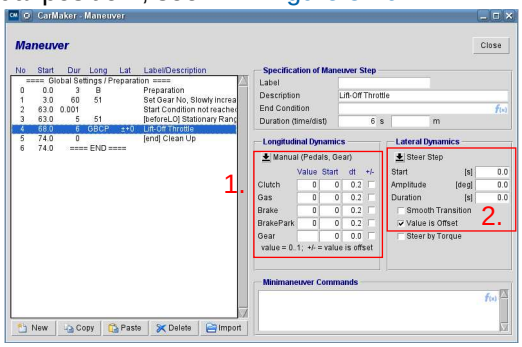


Figure 5.20: Manual interface for pedals and steering wheel

The TestRun ends with maneuver no. 5, it shows a "Clean Up" maneuver, because the duration is set to zero. This leads to a special maneuver if the TestRun is finished, no matter how the TestRun ends (normal termination, stop, error).

The command *DVArel* * in the minimaneuver commands resets all quantities which were overwritten by DVA command. See *CarMaker User's Guide section DVA release*.

LiftOffTurnIn

Please read TestRun LiftOffOversteer before, because it shows nearly the content of this testrun, except the difference in maneuver no. 4.

When the throttle is released in maneuver no. 4, by a manual maneuver, see Figure 5.21, the steering angle is also changed by a manual steer maneuver, see *Main GUI> Parameters > Maneuver.* The defined steering amplitude is performed additional to the actual steering angle (defined by the option "Value is Offset").

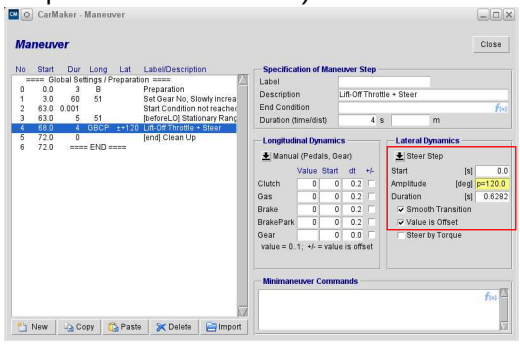


Figure 5.21: Difference of maneuver no.4, defined steering maneuver

LiftOff.ts

The TestSeries **LiftOff.ts** demonstrates how to perform a preparation test to identify a characteristic speed value and executing two TestRuns dependent of the identified value.

The series starts with the selection of a vehicle, see (1.) in Figure 5.22 and the first TestRun LiftOff_PreTest for determining the quantity "SteadyState.MaxSpeed". The TestRun is followed by a ScriptControl segment. The command sets the NamedValue "MaxSpeed". The quantity is accessible by the **get <QuantityName>** function.

```
NamedValue set MaxSpeed [format %.2f [expr 0.9*[get SteadyState.MaxSpeed]]]
```

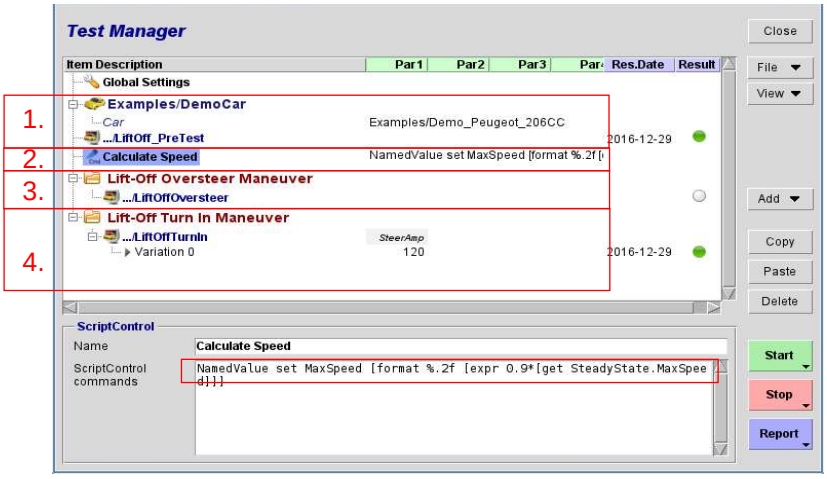


Figure 5.22: LiftOff series

In the 3rd section the TestRun LiftOffOversteer is performed. The value for the speed "MaxSpeed" was set by the previous script control command.

The 4th section shows the TestRun LiftOffTurnIn. This TestRun is also performed with the predefined NamedValue *MaxSpeed.* The amplitude of the turn in maneuver can be modified by the NamedValue *SteerAmp* which defines the additional amplitude for the steering maneuver of the TestRun.

RunOffTheRoad

The TestRun shows a rollover situation caused by running off the road, which is provoked by too high velocities and also imposed manual acceleration (gas pedal = 1) in the corner. The tilting moment of the car is reached by the high friction coefficient of the roadside.

The functions used in this TestRun are

- Friction Stripe in the "*Override selected Segment Attributes*" in the road segment user interface
- the preparation of the *"Start Values"* including starting velocity and gear number is set in the *Main GUI> Parameters > Maneuver > Global Settings*

Figure 5.23: Deviation from the road in addition with high friction coefficient of the roadside

Ramp

The TestRun shows a ramp created by the CarMaker function "Mesh".

The TestRun shows the CarMaker features

- creating drivable mesh objects
- maneuver end condition

The TestRun is finished if the End Condition of the maneuver is true, see following line. If the cars roll angle is greater than 78 degree the TestRun will be terminated.

```
abs(Vhcl.Roll)>rad(78)
```

The mesh object is not considered in the calculation of the "Static Desired Speed" hence the driver does not reduce the speed while driving against the ramp.

The drivable mesh object, see Figure 5.24 is developed by a point list in the *Main GUI> Parameters > Road > Segments > New Bump/Marker > Mesh* user interface. For more information about the mesh object see CarMaker User's Guide chapter Adding Road Bumps.

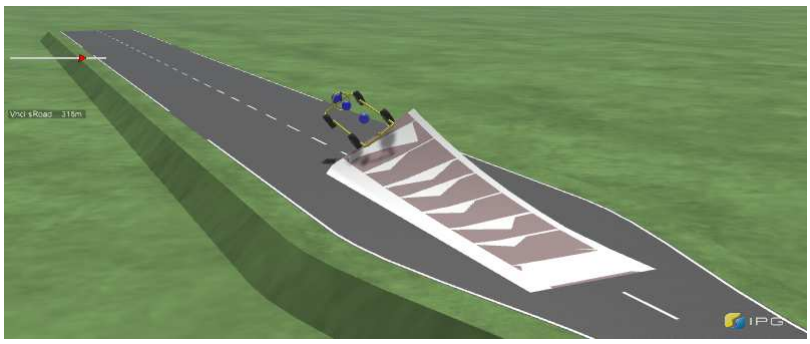


Figure 5.24: Mesh object and ABRAXAS vehicle model

MovingMass

This TestRun shows a vehicle rolling over as a consequence of unsecured cargo moving around. The cargo moves to the right side of the vehicle as it drives through a bend. Then, it bumps against the vehicles side. The force resulting from this impact is simulated in the second maneuver by applying a virtual force on the point of attack (Virtual.PoA). The point of attack is defined in the vehicle's *Additional Parameters*.

The TestRun is based on these CarMaker features

- Adding a load to the vehicle *(Main GUI> Parameters > Load),*
- Moving the load and applying a virtual force using *DVA Commands*

Initially, the vehicle drives with a velocity of 96 kph in sixth gear *("Main GUI> Parameters > Maneuver >* ***Global Settings****")*.

In maneuver no. 1 the position of the load is moved to a position of -0.7 m on the right side of the vehicle in 0.53 seconds.

```
DVAwr Car.Load.0.ty AbsRamp 530 -0.7 530
```

Then, the load "bumps against the vehicle's side panel". The effect of this impact is simulated using a virtual force command.

```
[DM.ManTime>=0.53] DVAwr Car.Virtual.Frc_1.y Abs 1 -146718
[DM.ManTime>=0.537] DVAwr Car.Virtual.Frc_0.y Abs 1 0
```

The virtual force is immediately applied at maneuver time 0.53s and set to zero again 0.007 seconds later.

SideImpact

The TestRun shows the vehicle hit by a side force in order to simulate a side impact with an other car.

The features used in this TestRun are

- setting the *Virtual Force* by DVA Command
- *using the Animation of virtual Forces, like in the TestRun* **VisualizedInteraction.**

The impact term is set by the maneuver duration of maneuver no.1. After the 0.1 s the Virtual Force is set to zero by DVA command

```
DVAwr Car.Virtual.Frc_1.y Abs 1 0
```

Fore more information about the virtual force and the DVA commands see CarMaker User's Guide.

5.6 Vertical Dynamics - Ride

Please note, that the IPG vehicle models are not validated beyond a range of 15Hz. For more detailled ride and comfort analysis in CarMaker, user defined vehicle models can be used instead.

Bumps

In real road traffic, bumps are usually used to reduce the driving velocity of the passing traffic. So vehicles have to deal with speed bumps and other overriding obstacles.

The TestRuns Bumps, BumpsUnsymmetrical and BumpsUnsymmetricalShifted shows several bumps with different heights and sizes. The driving speed will not be reduced because of the bumps. The bumps are created in the *Main GUI> Parameters > Road > Segment* user interface.

Further information about the different parameters are shown in CarMaker User's Guide chapter Adding Road Bumps.

The TestRun **Bumps** shows three consecutive bumps of the kind "Beam". The Bumps are created via the *Main GUI> Parameters > Road > Segments > New Bump/Marker* user interface. The drivers maneuver shows a single maneuver over 100 s and a target driving speed of 10 km/h.

BumpsUnsymmetrical

The TestRun **BumpsUnsymmetrical** shows six different bumps of the kind "Beam". The beams are arranged in pairs and form an asymmetrical obstacle. The beams have the same Start Offset (see Figure 5.25) and are shifted via the *y Offset* from the middle of the road. The maneuver shows a driving duration of 100 s with a velocity of 10 km/h.

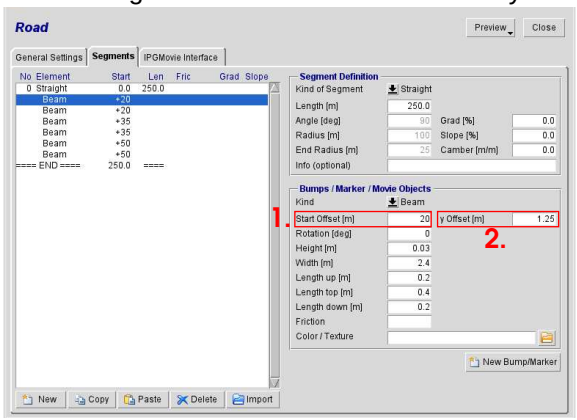


Figure 5.25: Road Bumps interface

BumpsUnsymmetricalShifted

The TestRun **BumpsUnsymmetricalShifted** is a variation of the TestRuns BumpsUnsymmetrical**.** The beams are split in the roads center line and are shifted in longitudinal direction (see Figure 5.26). The color of the beams can be modified via the input field "Color / Texture". The color-code must be entered in the following syntax "0.2,0.6,1.0,0.5" (red,green,blue,transparency) in the range of 0...1.

Figure 5.26: Shifted unsymmetrical beams

Waves

The TestRun shows the crossing of different road waves with variation in amplitudes and frequency. The waves are created in the *Main GUI> Parameters > Road > Segments > New Bump/Marker* user interface.

If the waves have a small amplitude, like in the first wave section of the TestRun, they are hardly visible in IPGMovie. The force to the wheel carrier shows the response to the three wave sections.

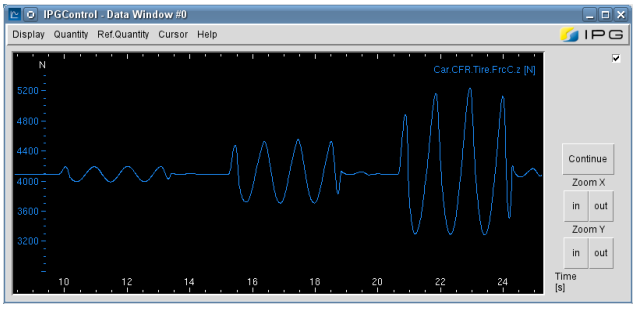


Figure 5.27: The quantity "Car.CFL.Tire.FrcC.z" shows the force response to the wheel carrier

WavesUnsymmetrical

The TestRun shows different wave sections on the right and left side of the road. The wave sections are created by the user interface *Main GUI> Parameters > Road > Segments > New Bump/Marker.*

More information about the Wave bumps are explained in *User's Guide* chapter *Adding Road Bumps* in section *Wave.*

Jumping

The TestRun shows a big jump over a hill only created with road segments. It demonstrates the numeric stability of the vehicle models.

The vehicle leaves the road surface. After a short flight duration it lands on a road segment with steep downhill gradient and drives on.

JumpingRamp

The TestRun shows a 60 m jump over a ramp.

Functions including in this TestRun are

- definition of drivable *Mesh* objects
- setting *Start Values* in the maneuver user interface

The ramps are created with a *Mesh* object in the road user interface. For detailed information see "Users Guide chapter *Adding Road Bumps* -Mesh".

Figure 5.28: Ramps created with *Mesh* object

Appendix A

List of Literature

[1] Proposal for a Council Directive amending Directive 70/220/EEC on the approximation of the laws of the Member States relating to measures to be taken against air pollution by emissions from motor vehicles, Official Journal of the European Communities, No C 81/1, 1990.

[2] André J. M.:The ARTEMIS European driving cycles for measuring car pollutant emissions, Science of the total environment, S. 334- 335 (2004) 73-84, 2004.

[3] André J. M., Lacour S., Hugot M., Oláh Z. and Joumard R.: Impact of the gearshift strategy on emission measurements, Artemis 3142 Report, IFSTTAR - LTE Laboratory Transports & Environment, France, 2003.

[4] N.N.: Safety standards for road vehicles Annex 42, Ministry of Land, Infrastructure, Transport and Tourism, Japan, 2009.

[5] N.N.: Measurement method notice Annex 41, Ministry of Land, Infrastructure, Transport and Tourism, Japan, 2008.

[6] N.N.: Code of Federal Regulations 40: Protection of Environment PART 86 ˘2013 Control of Emissions from new and in-use highway vehicles and engines, United States Government Printing Office, Washington, 1995.

[7] N.N.: www3.epa.gov, 19.04.2016

[8] Worldwide Harmonized Light Vehicles Test Procedure. Global Technical Regulation on Worldwide Harmonized Light Vehicles Test Procedure. ECE/TRANS/180/Add.15, 12.05.2014, www.unece.org (accessed 20.09.2016).

[9] NATIONAL HIGHWAY TRAFFIC SAFETY ADMINISTRATION, et al. FMVSS No. 126 electronic stability control systems. Final Regulatory Impact Analysis. March, 2007.

[10] SIMMERMACHER, Daniel. Objektive Beherrschbarkeit von Gierstörungen in Bremsmanövern. VDI Verlag, 2013, p. 179.

[11] Allgemeiner Deutscher Automobil-Club e.V. (https://www.adac.de): Der ADAC Ausweichtest - Nie ohne ESP, 2000.