Date:	2019-10-28
Author:	Sophie Schäferle, IPG Solutions Engineering
Release No.:	CM-8.0.2

How Can I Use IPGDriver in the Context of ADAS and AD Testing?

The IPGDriver model can be very independent. If not given any deviating orders, IPGDriver will shift gears, brake, accelerate and overtake on his own terms – and these are just a few points on the very long list of functionalities.

When simulating in the scope of ADAS and AD applications, we often want to support the driver's actions by having our control algorithms correct his driving behavior just slightly. Sometimes, however, we want the driver to neglect his usual tasks entirely so that our assistance systems can take over and we can test our controllers. In this article I want to show you a quick and easy way to set CarMaker's IPGDriver in a passive state.

Fields of Application

The different levels of ADAS and AD controllers are designed for different driving situations where the driver's behavior is critical. While ADAS systems are used to optimize the actions of an active driver, AD functionalities are developed for when the driver is inactive for a short or far longer period of time. Divided into the five levels of driving automation by SAE, the driver needs to be either absent for a short time (SAE Level 3) or entirely passive (SAE Level 4/5).

In CarMaker we have two possibility to simulate all situations. The simulation of an active driver is covered entirely by the IPGDriver model, both absent driver situations can be achieved by putting the IPGDriver model in a *passive* state for a specific amount of time.

Passive States for Level 4/5

The overall passive state is split into actions that are relevant for longitudinal movement (braking, accelerating) and lateral movement (steering). These can be controlled independent of one another or at the same time.

The passive state is described and set using the following two quantities: *Driver. Lat.passive, Driver.Long.passive.*

Both Quantities only have two possible states:

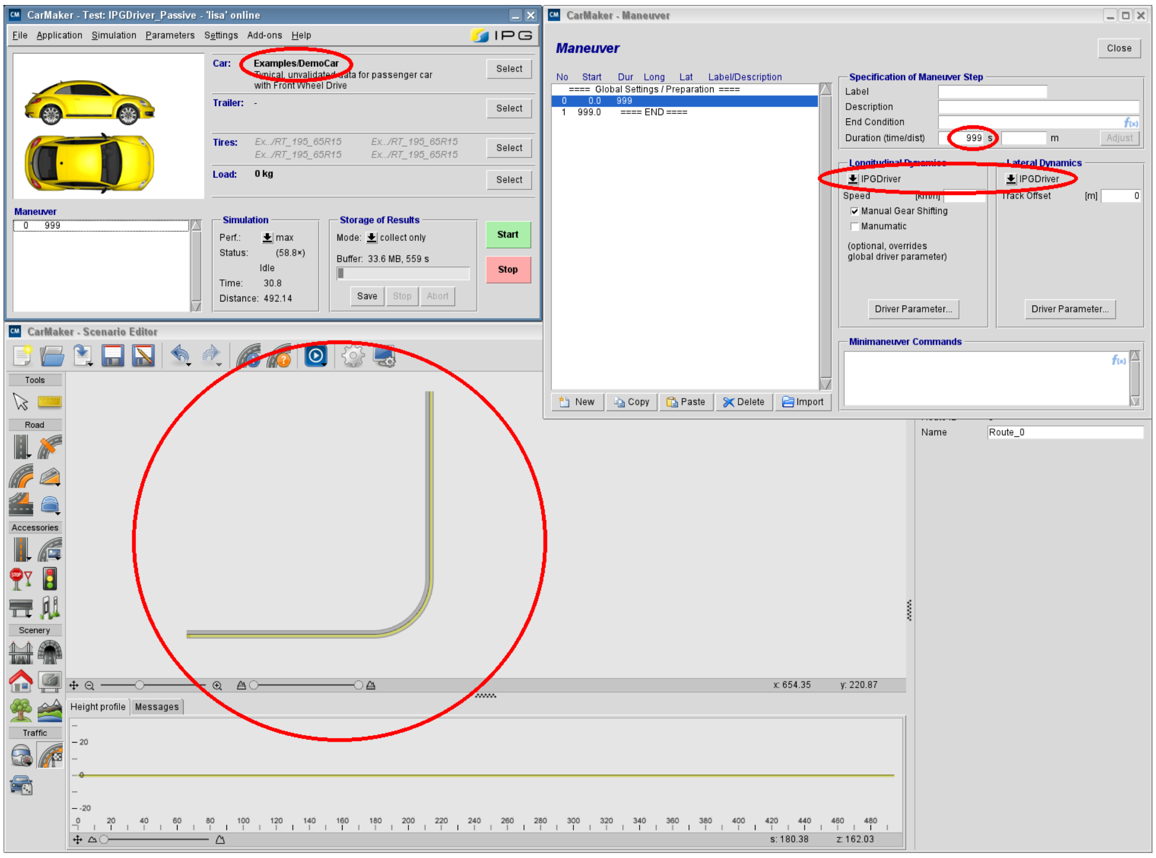
- 0, which means as much as "off" – the driver is not passive
- 1, which means as much as "on" – the driver is passive

There are different methods of manipulating these quantities (as there are for all writeable quantities), e.g. using DVA-Access, Real Time Expressions or writing to the quantity using an external tool. In this article I'd like to concentrate on using Real Time Expressions within a defined maneuver.

Use Case: Lateral Controller

In this example we're going to go with a use case where we want to test a lateral controller for AD applications that is to support the driver when he is temporarily distracted. First, I'm going to build a very simple TestRun:

- Vehicle: DemoCar
- Road: A straight segment, followed by a 90 degree turn and another straight segment
- Maneuver: One minimaneuver step with Duration = 999s, IPGDriver for both Longitudinal and Lateral Dynamics
- IPGDriver Cruising Speed = 50km/h



CarMaker - Test: IPGDriver_Passive - 'lisa' online

Car: Examples/DemoCar
Typical, unvalidated data for passenger car with Front Wheel Drive
Trailer: -
Tires: Ex../RT_195_65R15 Ex../RT_195_65R15 Ex../RT_195_65R15 Ex../RT_195_65R15
Load: 0 kg

Maneuver
0 999

Simulation
Perf.: max
Status: (58.8×) Idle
Time: 30.8
Distance: 492.14

Storage of Results
Mode: collect only
Buffer: 33.6 MB, 559 s

CarMaker - Maneuver

No	Start	Dur	Long	Lat	Label/Description
	==== Global Settings / Preparation ====				
0	0.0	999			
1	999.0	==== END ====			

Specification of Maneuver Step
Label:
Description:
End Condition:
Duration (time/dist): 999 s

Longitudinal Dynamics: IPGDriver
Speed [km/h]
☑ Manual Gear Shifting
☐ Manumatic
(optional, overrides global driver parameter)

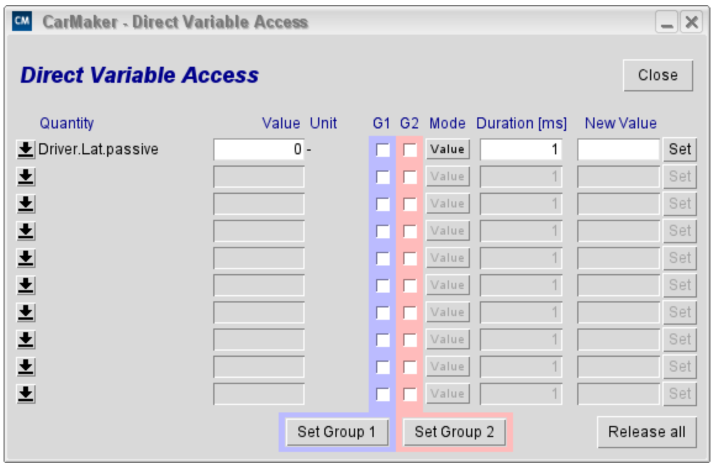
Lateral Dynamics: IPGDriver
Track Offset [m] 0

Minimaneuver Commands

CarMaker - Scenario Editor
Name: Route_0
x: 654.35 y: 220.87
s: 180.38 z: 162.03

Fig. 1. TestRun parameterization

When we start the simulation we can check the value of Driver.Lat.passive (interesting for us in this case) in the DVA Dialog (CarMaker Main GUI > Application > Direct Variable Access). Per default, this value is always 0.



CarMaker - Direct Variable Access

Direct Variable Access

Close

Quantity	Value	Unit	G1	G2	Mode	Duration [ms]	New Value	
Driver.Lat.passive	0	-	☐	☐	Value	1		Set
			☐	☐	Value	1		Set
			☐	☐	Value	1		Set
			☐	☐	Value	1		Set
			☐	☐	Value	1		Set
			☐	☐	Value	1		Set
			☐	☐	Value	1		Set
			☐	☐	Value	1		Set
			☐	☐	Value	1		Set
			☐	☐	Value	1		Set

Set Group 1   Set Group 2   Release all

Fig. 2. Default: Driver.Lat.passive = 0

Now we're going to use Real Time Expressions (RTE) to manipulate Driver.Lat.passive as to simulate a distracted driver whose vehicle slowly drifts off course. The trigger for our RTE is going to be distance based so that when the vehicle is approaching the turn, we can set the driver passive to simulate him not steering in time to take the turn with the correct angle. For this we'll use the quantity *Car.Road.sRoad* and set the trigger to *190m*. The command is setting *Driver.Lat.passive* to *1* (which means "Driver is passive"). After the turn, we'll unset the passive state.That leaves us with

Eval Car.Road.sRoad>190 ? Driver.Lat.passive=1
Eval Car.Road.sRoad>280? Driver.Lat.passive=0

Now if we start the simulation again we can see that after 190m travel the Driver stops acting on the steering wheel and therefore, leaves the road when he comes to the turn.

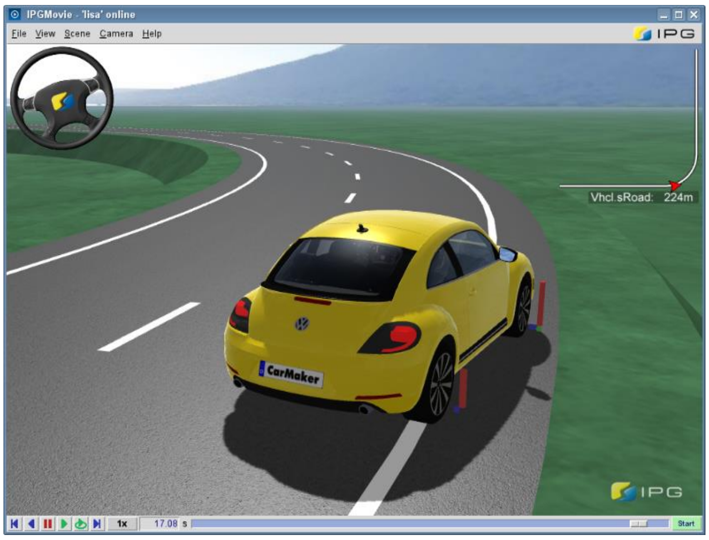
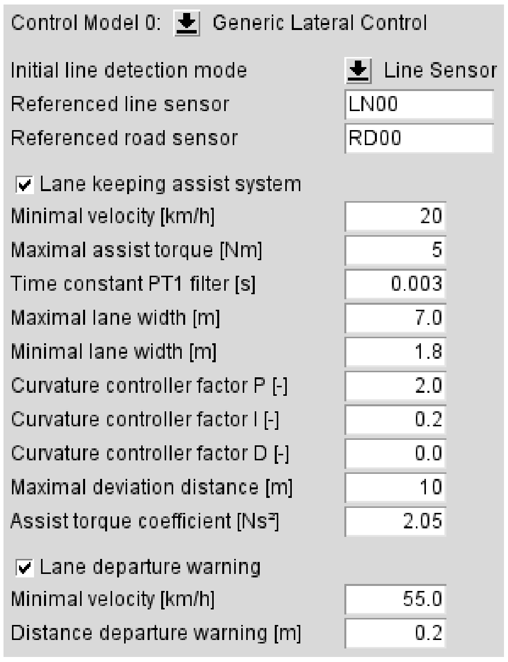


Fig. 3. Passive driver leaving the road

Not that we've simulated a distracted driver, we can add a controller to our vehicle that will interfere with the driver's actions and help keep him on course. In this example, I'm going to demonstrate using our example controller *Generic Lateral Control* which we can use to simulate a simple AD algorithm for lateral control.

In the *Vehicle Data* set under Vehicle Control I can add the Generic Lateral Controller and need to make sure that I also add a *Road Sensor*, *Line Sensor* and change the *steering* model to *GenTorque* for the controller to work. The controller itself is parametrized as follows

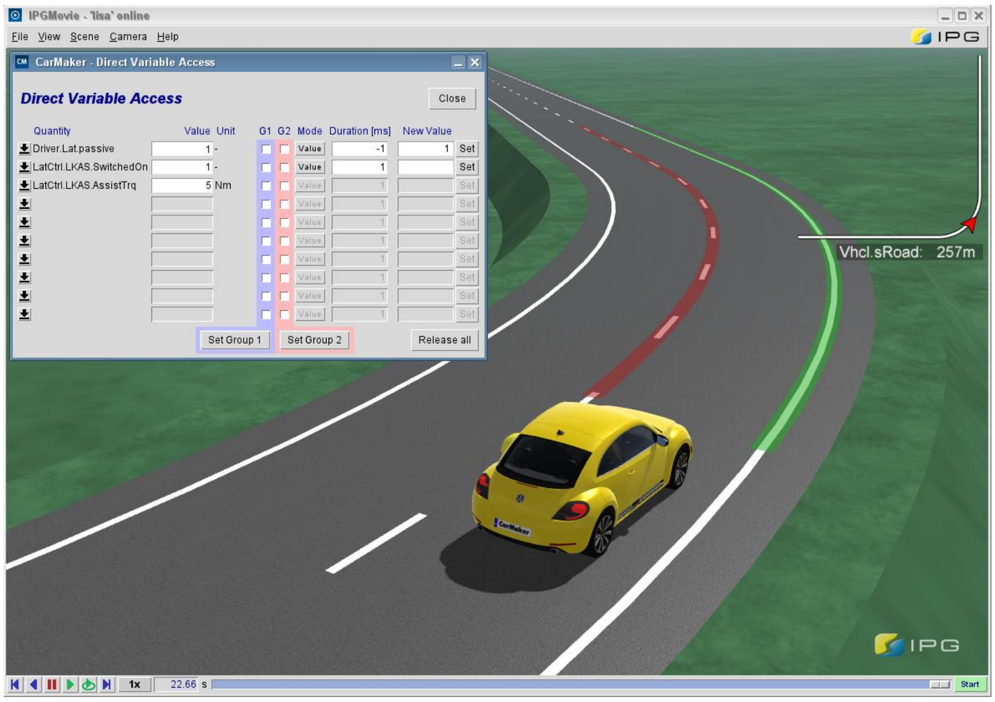


Control Model 0: Generic Lateral Control

Initial line detection mode	Line Sensor
Referenced line sensor	LN00
Referenced road sensor	RD00
✓ Lane keeping assist system	
Minimal velocity [km/h]	20
Maximal assist torque [Nm]	5
Time constant PT1 filter [s]	0.003
Maximal lane width [m]	7.0
Minimal lane width [m]	1.8
Curvature controller factor P [-]	2.0
Curvature controller factor I [-]	0.2
Curvature controller factor D [-]	0.0
Maximal deviation distance [m]	10
Assist torque coefficient [Ns²]	2.05
✓ Lane departure warning	
Minimal velocity [km/h]	55.0
Distance departure warning [m]	0.2

Fig. 4. Generic Lateral Controller parametrization

If we now start the simulation again and check the driver's state, the controllers state and the assisting torque of the controller in the DVA dialog, we can see that although the driver is inactive, our control system is keeping him on track.



IPGMovie - 'lisa' online

CarMaker - Direct Variable Access

Direct Variable Access

Quantity	Value	Unit	G1	G2	Mode	Duration [ms]	New Value	
Driver.Lat.passive	1	-			Value	-1	1	Set
LatCtrl.LKAS.SwitchedOn	1	-			Value	1		Set
LatCtrl.LKAS.AssistTrq	5	Nm			Value	1		Set

Vhcl.sRoad: 257m

22.66 s

Fig. 5. Lane Keeping Assistant taking over while IPGDriver is passive

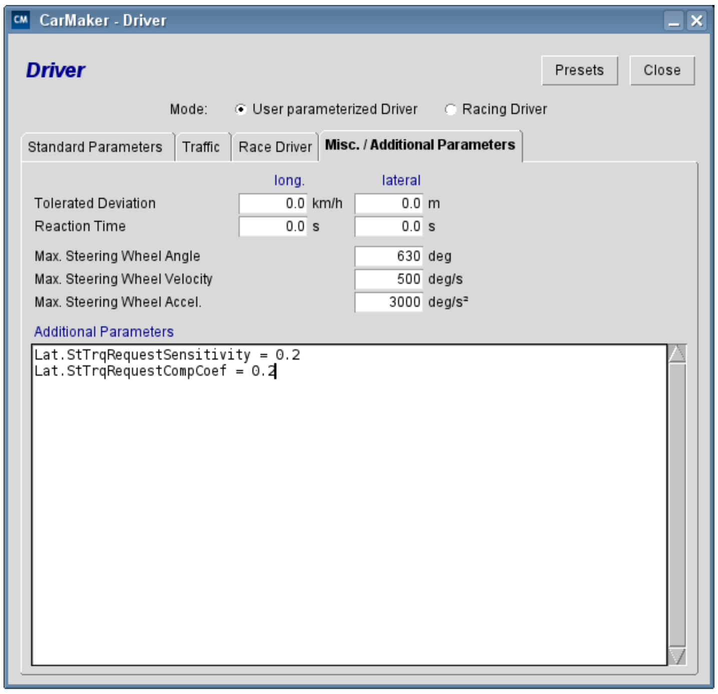
IPGDriver Additional Parameters for Level 1-3

In the example above, the vehicle controller takes over all steering actions for the time the driver is inactive. In other applications, where the driver's actions aren't being replaced, but only supported, we have two models running in parallel and interacting with one another: The IPGDriver and Vehicle Control model.

When the controller becomes active the driver finds himself in a state of transition where he is suddenly receiving external input. In the case of a lane keeping assistant this means that a steering torque is applied to the steering wheel in addition to the driver's own manual input. In reality, it's important to make sure this transition is as smooth as possible in order to not startle the driver with hasty movements and also not override his will completely. There's a way to consider this in CarMaker as well: Using *Additional Parameters*. These are available for fine-tuning the IPGDriver model. A list of these can be found in the IPGDriver manual, they are implemented in the *Misc. / Additional Parameters* tab of the Driver dialog. Two of these in particulare are relevant for our use case:

- *Lat.StTrqRequestSensitivity*: A value between 0 and 1 defining how tolerant the driver will be to external steering inputs; meaning *will he allow them*?
- *Lat.StTrqRequestCompCoef*: A value between 0 and 1 defining the driver's toleration of the new course selection by the external steering torque interaction.

These are implemented as shown in the screenshot below.



Mode: [✓ User parameterized Driver] Racing Driver

	long.		lateral	
Tolerated Deviation	0.0	km/h	0.0	m
Reaction Time	0.0	s	0.0	s
Max. Steering Wheel Angle			630	deg
Max. Steering Wheel Velocity			500	deg/s
Max. Steering Wheel Accel.			3000	deg/s²

Additional Parameters

```
Lat.StTrqRequestSensitivity = 0.2
Lat.StTrqRequestCompCoef = 0.2
```

Fig. 6. IPGDriver Additional Parameters

Try varying the parameters and having a look at the slight changes in driving behavior in IPGControl and IPGMovie (Hint: Look at the *Primary and Reference Vehicle* functionality in IPGMovie).