Samples Pack

ADAS_LKA_ACC

TABLE OF CONTENTS

1. INTRODUCTION 2
1.1 DESCRIPTION 2
2. GET STARTED 3
2.1 USING THE COMPILED MODULES 3
2.2 USING THE SIMULINK MODEL INSTEAD 5
3. EXPLANATIONS 7
3.1 GENERAL PRINCIPLE 7
3.2 SENSORS 7
3.3 DRIVER COMMAND 8
3.4 VEHICLE CONTROL 9
3.5 SIMULINK 9

1. INTRODUCTION

1.1 Description

This sample demonstrates the use of ScanerAPI and other SCANeR™ features to connect external ADAS algorithms to a SCANeR™ simulation.

Two simplified algorithms are used:

- ADAS_LKA (lane keeping assist, available in C only)
- ADAS_ACC (active cruise control, available in C and Simulink)

During the simulation, both functions can be activated independently using the keyboard. Effects can be observed on the main visual. A separate window displays the current status of ADAS the functions.

2. GET STARTED

2.1 Using the compiled modules

→ Select configuration EVAL_19_ADAS_LKA_ACC
CONFIGURATION > Configuration Manager… > EVAL_19_ADAS_LKA_ACC

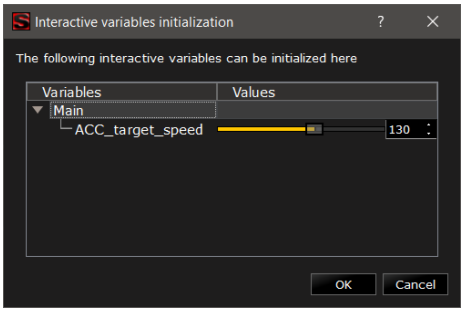
The necessary modules start automatically.

ⓘ *Notice the custom modules ADAS_ACC and ADAS_LKA, among the other standard modules.*

→ Open the scenario straight_lka_acc.sce 

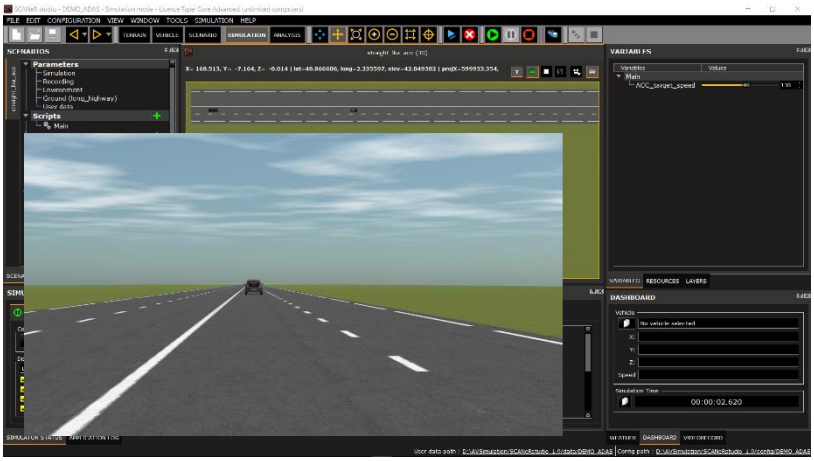
→ Start the simulation 

Set the default ACC target speed



→ Press OK

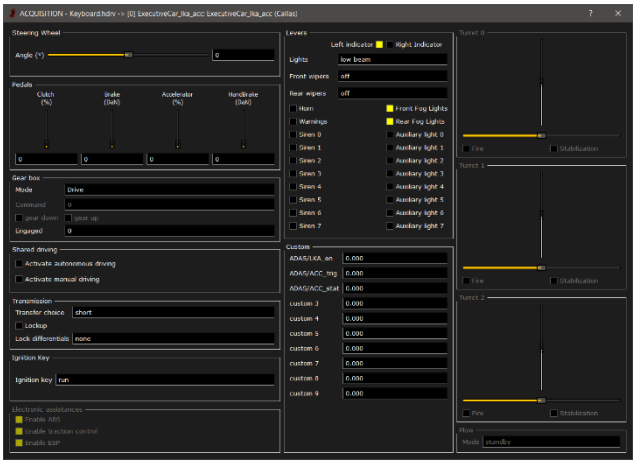
The simulation starts



→ The EGO vehicle can be driven with keyboard ⇧ ⇩ ⇦ ⇨

The TARGET vehicle drives at constant speed (90km/h)

ⓘ *If the EGO ignition key is not "run", restart MODELHANDLER*



→ Toggle LKA with F1

The state of the LKA is in the Controlpad Viewer

The steering wheel turns towards the middle of the lane when the car diverges.

→ Toggle ACC with F2

States are: OFF → 20m → 30m → 40m → 50m → OFF

The car reaches the target speed (set at the beginning of the simulation, 130km/h by default) – or it stays behind the TARGET vehicle if it is too close.

A graph of the distance to collision, as seen by the ACC, is drawn in the Controlpad Viewer

ⓘ *ACC is meant to work close to (20m to 100m), and at similar speed (±20kmph) compared to the TARGET vehicle. Use the keyboard to reach these conditions.*

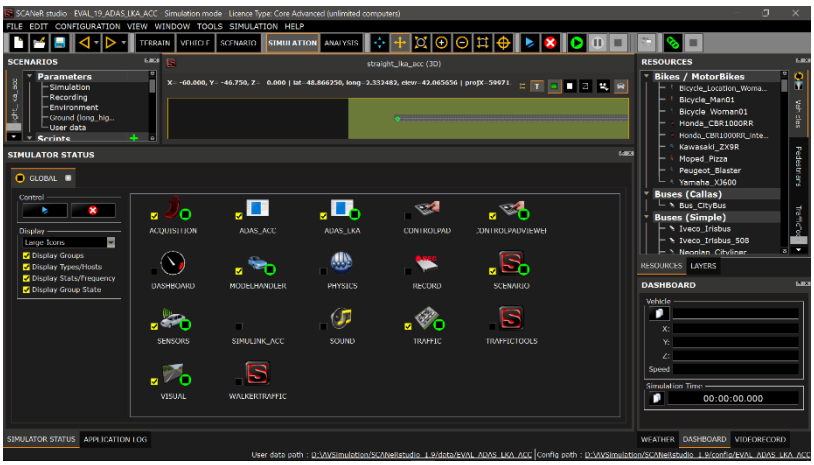
2.2 Using the Simulink model instead

A Simulink model is provided that does the same as the compiled C module ADAS_ACC.

⚠ MATLAB r2016b is necessary for this part.

→ Stop the simulation

→ Stop the ADAS_ACC module
Right click > Stop process *or double-click on it.*



→ Open MATLAB r2016b

- Navigate to %STUDIO_PATH%/SCANeRstudio_2021/APIs/
- Run the MATLAB script setupSCANeR.m
- Navigate to ./samples/ScanerAPI/ADAS/adas_acc_simulink/
- Open the Simulink model adas_acc_simulink.mdl
- Start the Simulink simulation.

[image missing]

The module SIMULINK_ACC is now tagged as in state "Daemon" , meaning it is ready to participate to the simulation.

→ Start the simulation

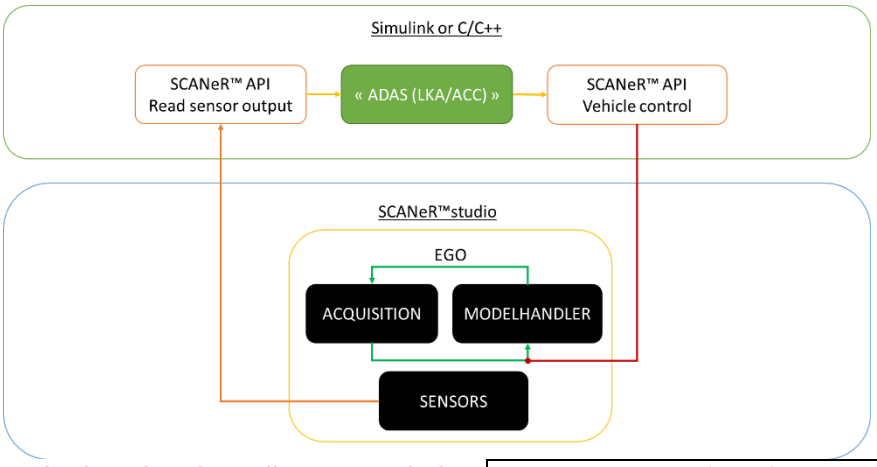
From there the behaviour is the same as with the compiled module ADAS_ACC.

3. EXPLANATIONS

3.1 General principle

The custom modules ADAS_LKA and ADAS_ACC are developed in C. Using the ScanerAPI, they get access to the simulation data shared on the SCANeR™ Network.

They both follow the same architecture:



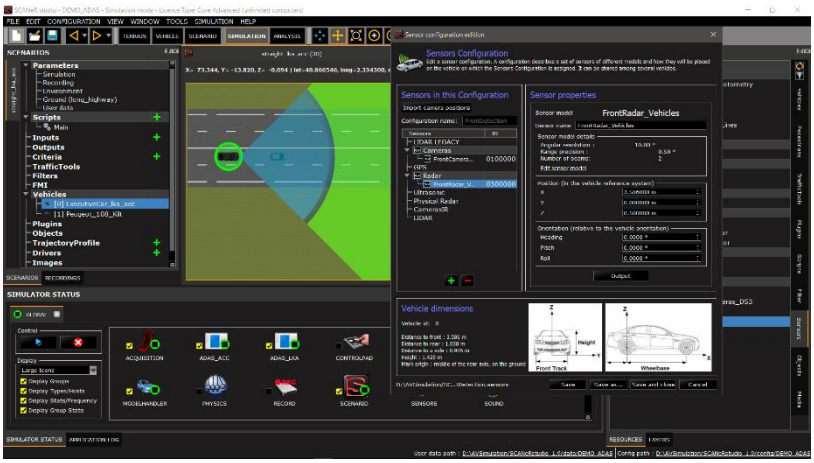
The C sources are in the Visual Studio 2013 solution %STUDIO_PATH%/APIs/samples/evaluation.sln (projects “adas_lka” and “adas_acc”).

Documentation for the ScanerAPI
(2.2. SCANeR API)

3.2 Sensors

The EGO vehicles is set to carry two sensors.

- A front CAMERA sensor detects the lanes and lanes (used by the LKA)
- A front RADAR sensor detects the TARGET vehicle (used by the ACC)



Documentation for the Sensors
(6.7.1.6. Sensors)

The custom modules ADAS_LKA and ADAS_ACC access the sensor outputs messages available on the SCANeR™ Network.

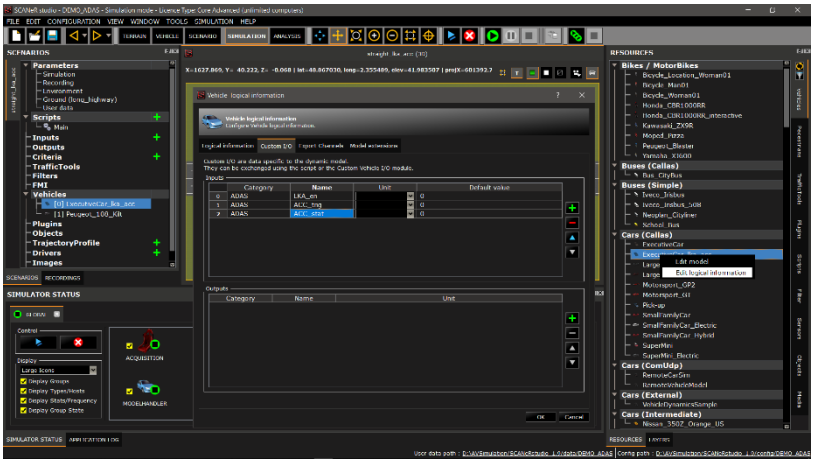
- ADAS_LKA reads the ISensor/RoadLinesPoints message from the CAMERA
- ADAS_ACC reads the ISensor/SensorMovableTargets message from the RADAR

Documentation for the Network messages
(Network.html)

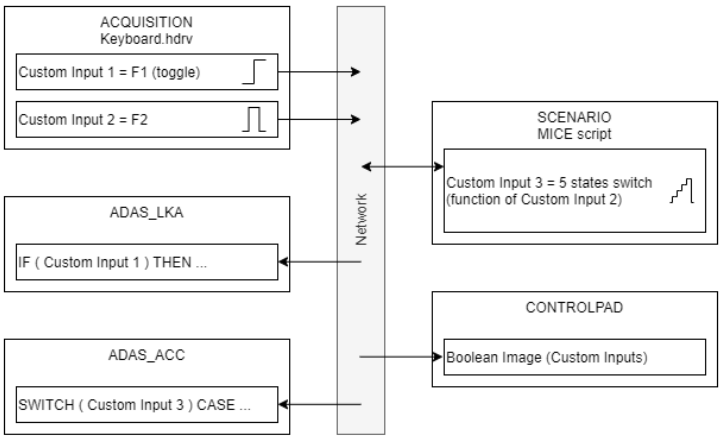
3.3 Driver command

The command status (on/off) of the ADAS features are assigned to Custom Inputs.

This is one way to formalize the data exchange between the modules on the SCANeR™ Network. The other options include the more generic Export Channels or VEN messages, preferred when no application-specific interface exist in SCANeR™.



- ACQUISITION assigns the "Fn" keys to the custom inputs by default.
  FnX→Custom Input *X*
  A hit on the F1 or F2 keys will change the value of the Custom Inputs 1 and 2.
- The MICE script accesses and modifies the Custom Inputs.
  While ACQUISITION only set binary states, the script adds more states in Custom Input 3.
- ADAS_LKA and ADAS_ACC read the values of the Custom Inputs using the SCANeR™ API
  The ADAS functions are activated depending of the Custom inputs 1 and 3.
- CONTROLPAD reads the Custom Inputs to display their values.



Documentation for the Custom Inputs
(5.5.2.1.2. Custom inputs / outputs)

3.4 Vehicle control

For performance issues, SCANeR™ uses a Shared Memory (Shm), instead of the Network, to manage the data related to advanced vehicles dynamics (*i.e.* Callas).

Documentation for the Vehicle command overload
(2.2.6.3.4.3. Overload the data from Shared Memory)

- Lateral control
  ADAS_LKA applies additional torque on the steering wheel to get lateral control.
  ModelCabin/SteeringToModelCorrective

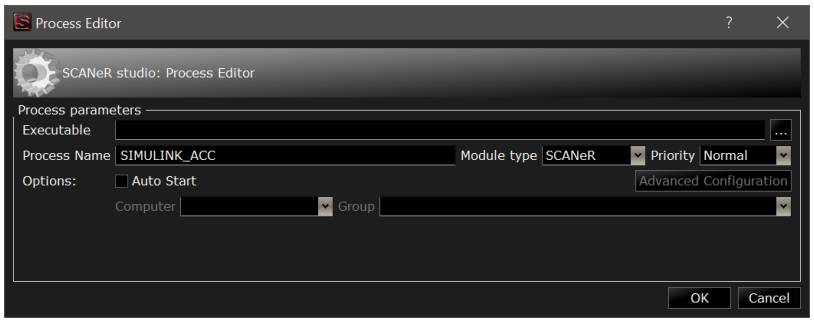
- Longitudinal control
  ADAS_ACC change the pedal commands (throttle and brake) to reach the target speed.
  ModelCabin/CabToModelCorrective

Documentation for the Shm messages
(shm.html)

3.5 Simulink

The Simulink model provided clones the code of ADAS_ACC with standard Simulink Blocks. It reads and write the exact same messages; hence it behaves the same.

- In SCANeR™ Studio, it is materialized as a dummy module with no executable attached.



- In Simulink, the “Controller” block materialize the link to SCANeR™.

[image missing]

Documentation for Simulink Co-Simulation
(2.3.5. Co-simulation)