Date:	2019-06-17
Author:	Pooja Hegde, Test Systems and Engineering, Germany
Release No.:	CM-7.1.1

Data Files

The CarMaker_CANoe_CoSim.zip file following contains the files.

1) CarMaker Project file: Created using the CarMaker GUI containing the general folder structure
2) CANoe Files: Contains the config file to be used, the database file that is needed, CAPL script used for controller logic and the final resulting FMU that is exported

Connecting CarMaker with CANoe through FMU

This is an example to connect CarMaker with CANoe for a Rest Bus Simulation of a generic ACC system. The connection is established to communicate signals between CarMaker and CANoe when both the tools are open and simulating simultaneously.

Technical Background

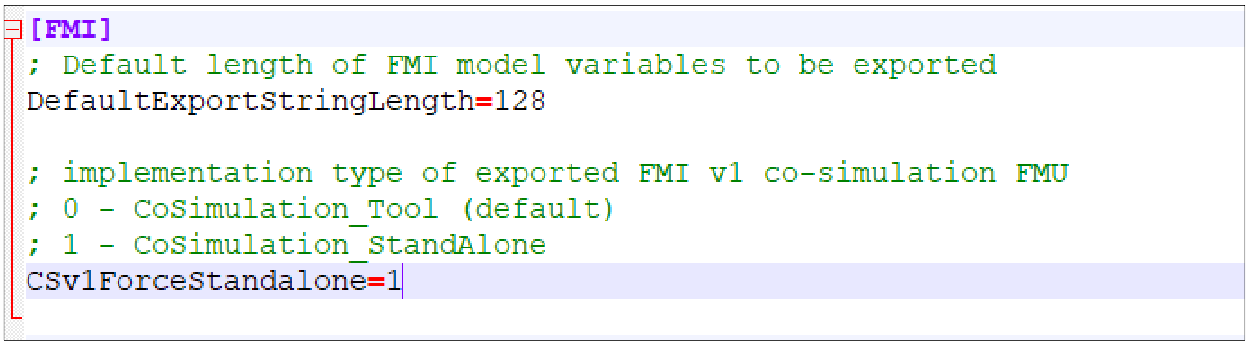
FMUs are used to exchange models conveniently between different tools directly as black box (or grey box) models. This example is based on FMU standard 1.0. The aim here is to obtain input signals to the FMU from CarMaker, then use these signals in CANoe where the ACC function is implemented and finally to give the output from the FMU ack to CarMaker. The controller is a very basic open loop proportional gain speed control since the aim of the example is to establish the connection between the tools and not develop the function.

Note: The officially supported FMU mode by CarMaker is of the type Co-simulation Standalone (Refer FMI for Co-Simulation). The default mode of the FMU from CANoe is set at Co-simulation Tool. A workaround is used to manually set the FMU mode to standalone even when both tools are open and running and hence this particular example is not officially supported.

CarMaker version: 7.1.1
CANoe version: 11.0

Setting the Simulation Mode

- Go to folder C:\ProgramData\Vector\CANoe\<version> and open file CAN.ini in a text editor
- Change the value of ***CSv1ForceStandalone*** to 1. This will set the configuration to be of type Co-simulation standalone



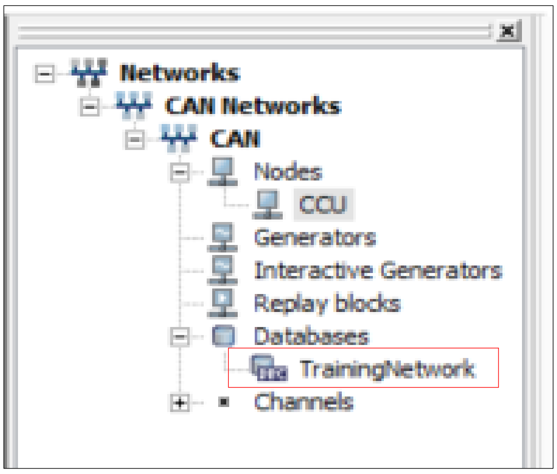
```
[FMI]
; Default length of FMI model variables to be exported
DefaultExportStringLength=128

; implementation type of exported FMI v1 co-simulation FMU
; 0 - CoSimulation_Tool (default)
; 1 - CoSimulation_StandAlone
CSv1ForceStandalone=1
```

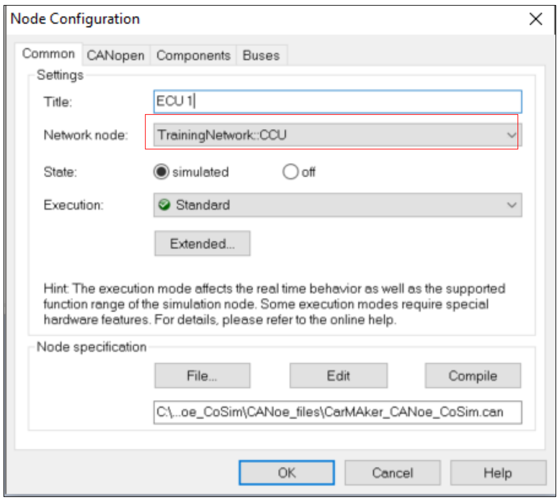
Setup: CANoe Components

All the required settings are in the Configuration file that is in this example. Below is a brief explanation of the components.

- Node: Nodes are added to the CAN network to simulate different ECUs
- Dbc file: Database file contains the raw signals that are used for data transfer
    - Drag and drop the dbc file from a folder into CANoe GUI. It will be automatically added to the network



- A database file needs to be added to the Node to access different signals
    - Right click on node and click on configure
    - Select the dbc file along with the required network node
    - In the components tab add the dll files required by the CAPL script that should be linked to the node. The filepath for the dlls is C:\Program Files\Vector CANoe 11.0\Exec32



Settings

Title: ECU 1

Network node: TrainingNetwork::CCU

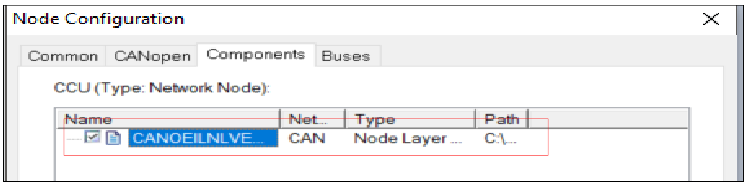
State: [✓ simulated] off

Execution: Standard

Hint: The execution mode affects the real time behavior as well as the supported function range of the simulation node. Some execution modes require special hardware features. For details, please refer to the online help.

Node specification

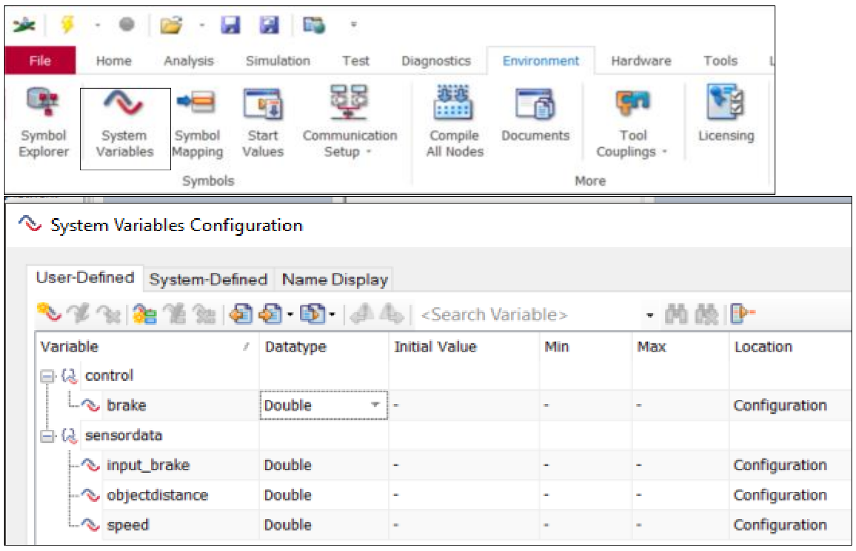
C:\...oe_CoSim\CANoe_files\CarMAker_CANoe_CoSim.can



CCU (Type: Network Node):

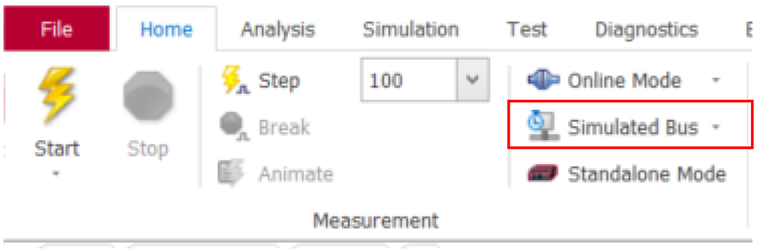
Name	Net...	Type	Path
CANOEILNLVE...	CAN	Node Layer ...	C:\...

- CAPL: Vector specific programming language that can be used to manipulate and calculate signals. This can be accessed by doing a right click on the node and selecting edit. Each node can have a separate CAPL script for different logics.
- System Variables: User defined variables to store and calculate data. Different datatypes can be assigned to the variables.
    - These variables can be either mapped to raw signals to store data or the user can initialize his/her own variables for internal usage in a script (as a trigger for Example)



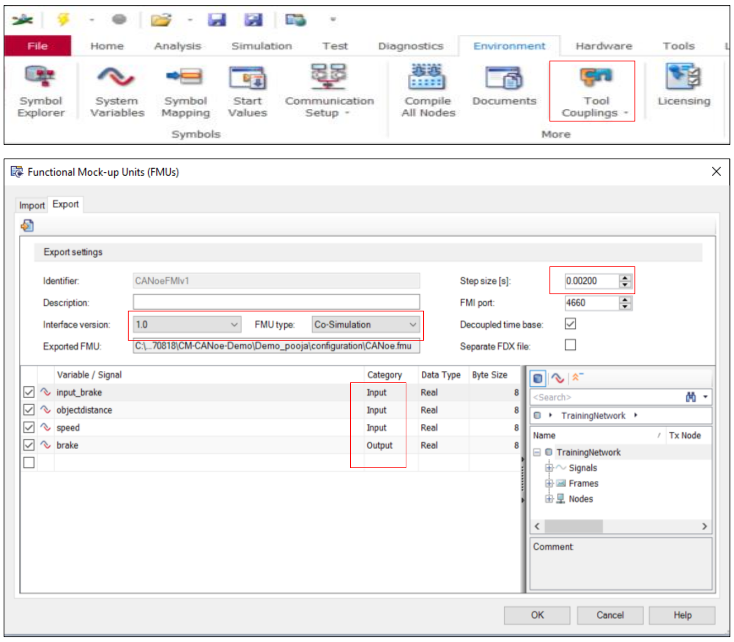
Variable	Datatype	Initial Value	Min	Max	Location
control					
brake	Double	-	-	-	Configuration
sensordata					
input_brake	Double	-	-	-	Configuration
objectdistance	Double	-	-	-	Configuration
speed	Double	-	-	-	Configuration

- Simulated bus: Select the Simulated bus option from the Home menu



FMU Export settings

- Go to Environment Tab-> Tool Couplings-> Functional Mockup Interface
    - Settings:
        - Interface Version:1
        - FMU type: Co-Simulation
        - Stepsize: 1ms (or other depending on simulation and datatypes)
- Be sure to assign the correct Input/Output status to each signal/variable
- Click on the export symbol on the top-left corner to export the FMU



Export settings

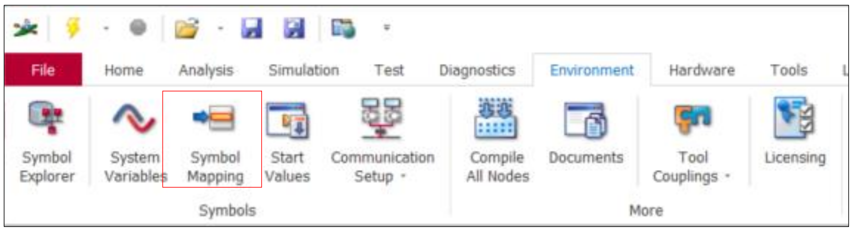
Identifier:	CANoeFMIv1
Description:	
Interface version:	1.0
FMU type:	Co-Simulation
Exported FMU:	C:\...70818\CM-CANoe-Demo\Demo_pooja\configuration\CANoe.fmu
Step size [s]:	0.00200
FMI port:	4660
Decoupled time base:	✓
Separate FDX file:	☐

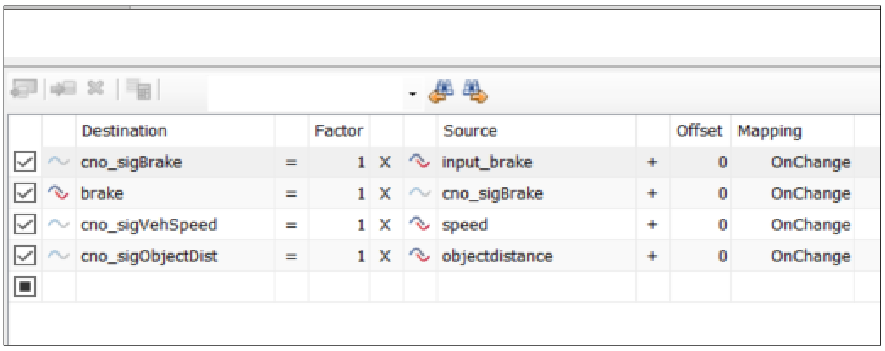
	Variable / Signal	Category	Data Type	Byte Size
✓	input_brake	Input	Real	8
✓	objectdistance	Input	Real	8
✓	speed	Input	Real	8
✓	brake	Output	Real	8
☐				

CANoe signal mapping

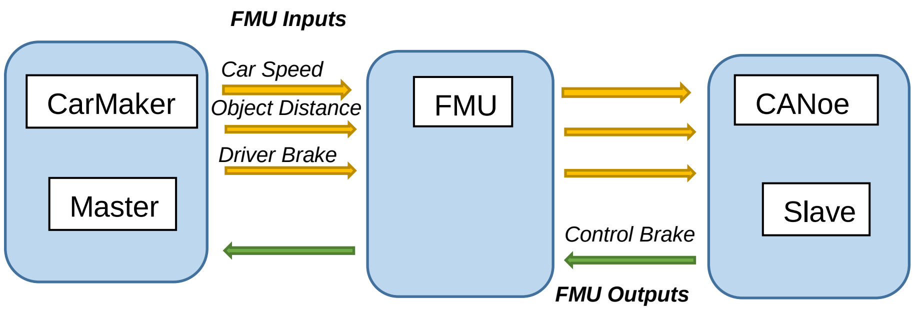
- Data inside CANoe has to be mapped to appropriate channels
    - Raw Signal data/data stored in variables should be mapped from a source on to a destination.
        - *Input_brake, speed* and *objectdistance* are coming in from CarMaker. Let us consider *input_brake*
        - It is first written into the raw signal *cno_sigBrake* from the dbc file
        - Some calculations are performed and this data is modified
        - This modified data is stored in the variable *brake*

   - This is the signal that will be sent back to CarMaker as a control input
- Note that it is important that the data types of the mapped signals match for the simulation to run without errors.
-





	Destination		Factor		Source		Offset	Mapping
✓	cno_sigBrake	=	1	X	input_brake	+	0	OnChange
✓	brake	=	1	X	cno_sigBrake	+	0	OnChange
✓	cno_sigVehSpeed	=	1	X	speed	+	0	OnChange
✓	cno_sigObjectDist	=	1	X	objectdistance	+	0	OnChange



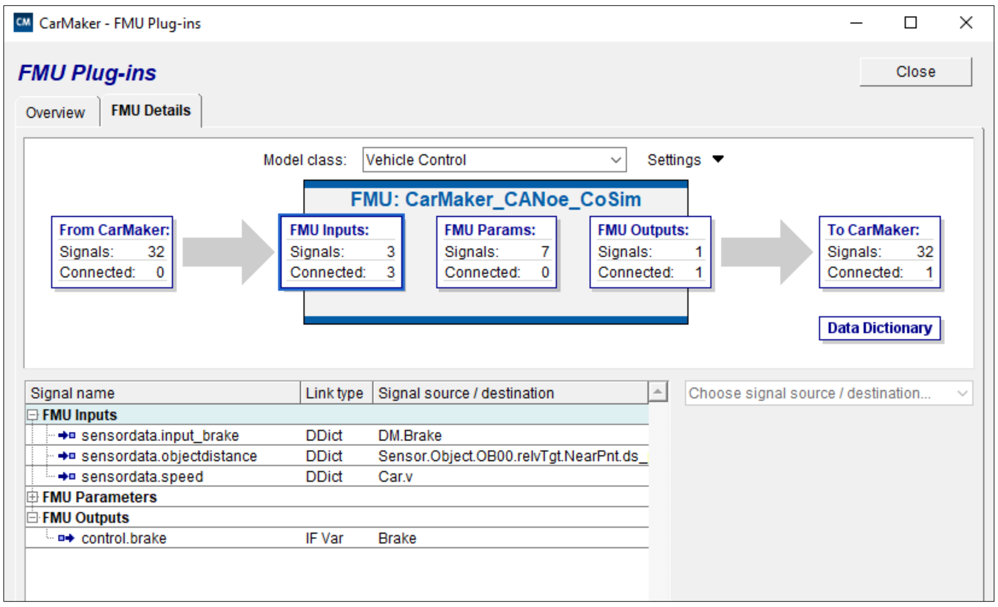
Signals

- Input_brake: current brake pedal position of the driver (DM.Brake)
- Objectdistance: distance to object nearest point (Sensor.Object.OB00.relvTgt.NearPnt.ds_p)
- Speed: Current vehicle speed (Car.v)
- Brake: Control signal for vehicle control (VC.Brake)

CarMaker signal mapping

Follow the usual FMU integration procedure

- Place exported FMU in plugins folder of the project directory
- Go to FMI GUI from Applications>FMU Plug Ins and update list
- Select the appropriate model class (Vehicle control in this example) and connect all the required FMU signals
- Replace the required subsystem model with FMU in the vehicle data set
- Start simulation



CarMaker - FMU Plug-ins

FMU Plug-ins

Overview | FMU Details

Model class: Vehicle Control

Settings

FMU: CarMaker_CANoe_CoSim

	Signals	Connected
From CarMaker:	32	0
FMU Inputs:	3	3
FMU Params:	7	0
FMU Outputs:	1	1
To CarMaker:	32	1

Data Dictionary

Signal name	Link type	Signal source / destination
FMU Inputs		
sensordata.input_brake	DDict	DM.Brake
sensordata.objectdistance	DDict	Sensor.Object.OB00.relvTgt.NearPnt.ds_
sensordata.speed	DDict	Car.v
FMU Parameters		
FMU Outputs		
control.brake	IF Var	Brake

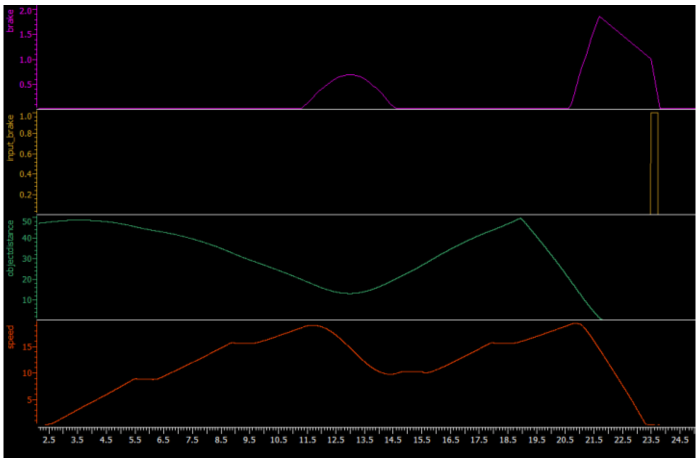
Choose signal source / destination...

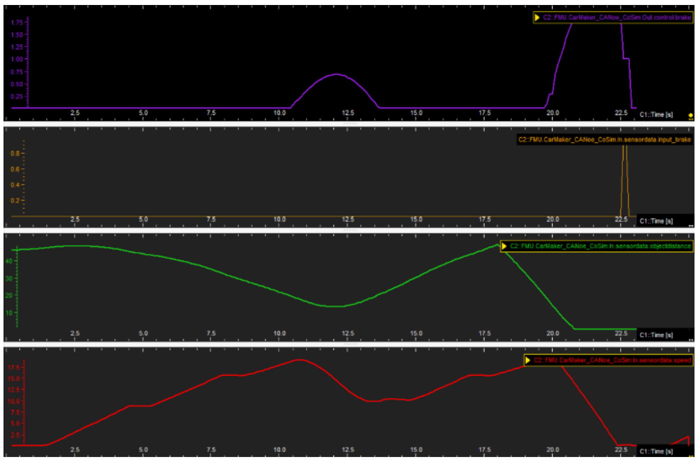
Note:

- All signals may not appear in the data dictionary of FMI GUI in the beginning. Run the testrun briefly once to update the list of CarMaker signals.
- The Driver setting has to be set so that the driver does not consider traffic. Otherwise the driver model will interfere with the controller built in the example.

Simulation Visualization

Comparing signals with IPGControl and CANoe-Graphics tools





CANoe-Graphics tool CarMaker-IPGControl