

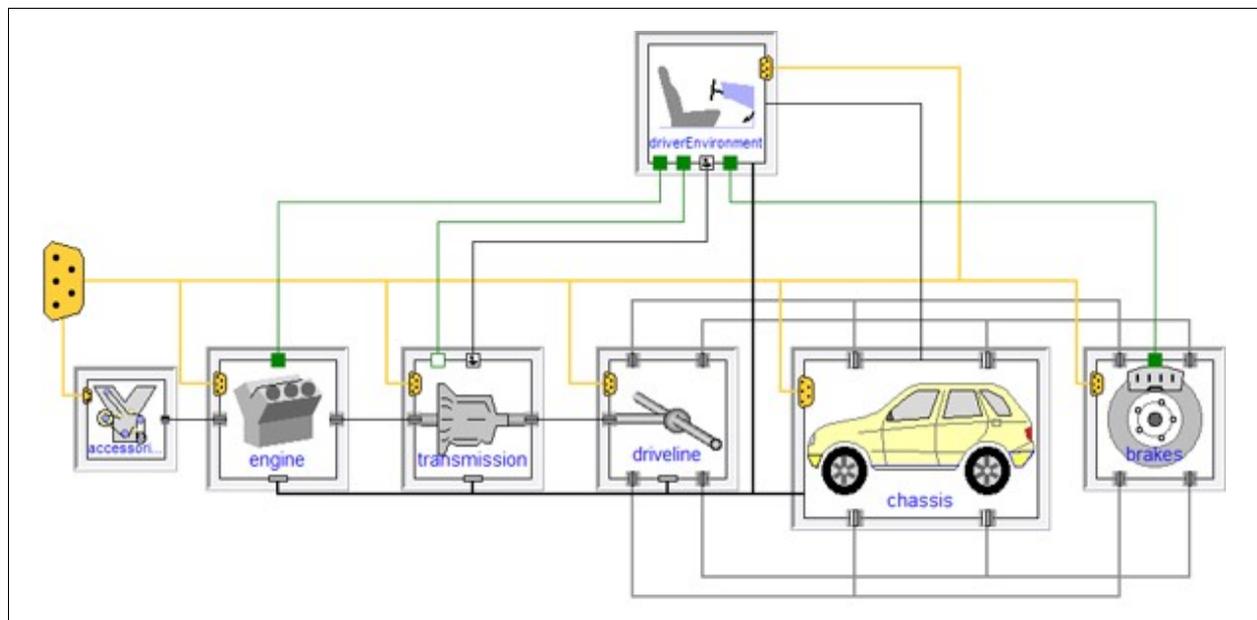


VehicleInterfaces Library

Version 1.2

August 2011

Tutorial and Reference



Copyright © 1999-2011, Dassault Systèmes, DLR, and Modelon. All rights reserved.

Dymola® is a registered trademark of Dassault Systèmes AB.

Modelica® is a registered trademark of the Modelica Association.

Other product or brand names are trademarks or registered trademarks of their respective holders.

Contents

VehicleInterfaces.....	8
VehicleInterfaces.UsersGuide.....	9
VehicleInterfaces.UsersGuide.QuickStart.....	11
VehicleInterfaces.UsersGuide.NamingConventions.....	12
VehicleInterfaces.UsersGuide.SignalBus.....	13
VehicleInterfaces.UsersGuide.SignalBus.AddingSignals.....	14
VehicleInterfaces.UsersGuide.DriverInteractionBus.....	16
VehicleInterfaces.UsersGuide.ModelRotatingParts.....	17
VehicleInterfaces.UsersGuide.SubsystemDefinitions.....	18
VehicleInterfaces.UsersGuide.Tutorials.....	18
VehicleInterfaces.UsersGuide.Tutorials.Tutorial1.....	18
VehicleInterfaces.UsersGuide.ReleaseNotes.....	20
VehicleInterfaces.UsersGuide.ReleaseNotes.Version_1_2.....	20
VehicleInterfaces.UsersGuide.ReleaseNotes.Version_1_1_2.....	20
VehicleInterfaces.UsersGuide.ReleaseNotes.Version_1_1_1.....	20
VehicleInterfaces.UsersGuide.ReleaseNotes.Version_1_1.....	21
VehicleInterfaces.UsersGuide.ReleaseNotes.Version_1_0.....	21
VehicleInterfaces.UsersGuide.Acknowledgements.....	22
VehicleInterfaces.UsersGuide.License.....	22
VehicleInterfaces.Examples.....	22
VehicleInterfaces.Examples.ConventionalAutomaticVehicle.....	23
VehicleInterfaces.Examples.ConventionalVehicle_AltNames.....	24
VehicleInterfaces.Examples.RearWheelDriveAutomaticVehicle.....	24
VehicleInterfaces.Examples.ConventionalAutomaticVehicleExternalDriver.....	25
VehicleInterfaces.Examples.ConventionalManualVehicle.....	25
VehicleInterfaces.Examples.FrontWheelDriveManualVehicle.....	26
VehicleInterfaces.Examples.PowerSplitHybrid.....	27
VehicleInterfaces.Examples.SeriesHybrid.....	27
VehicleInterfaces.Examples.PartialVehicle.....	28
VehicleInterfaces.Blocks.....	28
VehicleInterfaces.Blocks.RealPassThrough.....	29
VehicleInterfaces.Blocks.IntegerPassThrough.....	29
VehicleInterfaces.Blocks.BooleanPassThrough.....	29
VehicleInterfaces.Blocks.InvertNormalizedInput.....	30
VehicleInterfaces.Icons.....	30
VehicleInterfaces.Icons.VariantLibrary.....	31
VehicleInterfaces.Icons.BaseClassPackage.....	31
VehicleInterfaces.Icons.Accessories.....	31
VehicleInterfaces.Icons.Atmosphere.....	32
VehicleInterfaces.Icons.Battery.....	32
VehicleInterfaces.Icons.Brakes.....	32
VehicleInterfaces.Icons.Chassis.....	32
VehicleInterfaces.Icons.Controller.....	32
VehicleInterfaces.Icons.DataDictionary.....	32
VehicleInterfaces.Icons.Driveline.....	32
VehicleInterfaces.Icons.DriverEnvironment.....	33
VehicleInterfaces.Icons.Driver.....	33
VehicleInterfaces.Icons.ElectricMotor.....	33
VehicleInterfaces.Icons.Empty.....	33
VehicleInterfaces.Icons.Engine.....	33
VehicleInterfaces.Icons.MultipleMounts.....	33
VehicleInterfaces.Icons.Road.....	34
VehicleInterfaces.Icons.SingleMount.....	34
VehicleInterfaces.Icons.TwoMounts.....	34

VehicleInterfaces.Icons.Transmission.....	34
VehicleInterfaces.Icons.SignalSubBusWithExplicitSignals.....	34
VehicleInterfaces.Interfaces.....	34
VehicleInterfaces.Interfaces.ControlBus.....	35
VehicleInterfaces.Interfaces.AccessoriesBus.....	36
VehicleInterfaces.Interfaces.AccessoriesControlBus.....	37
VehicleInterfaces.Interfaces.BatteryBus.....	37
VehicleInterfaces.Interfaces.BatteryControlBus.....	37
VehicleInterfaces.Interfaces.BrakesBus.....	37
VehicleInterfaces.Interfaces.BrakesControlBus.....	37
VehicleInterfaces.Interfaces.ChassisBus.....	38
VehicleInterfaces.Interfaces.ChassisControlBus.....	38
VehicleInterfaces.Interfaces.DrivelineBus.....	38
VehicleInterfaces.Interfaces.DrivelineControlBus.....	38
VehicleInterfaces.Interfaces.DriverBus.....	38
VehicleInterfaces.Interfaces.DriverInterface.....	39
VehicleInterfaces.Interfaces.ElectricMotorBus.....	39
VehicleInterfaces.Interfaces.ElectricMotorControlBus.....	39
VehicleInterfaces.Interfaces.EngineBus.....	39
VehicleInterfaces.Interfaces.EngineControlBus.....	39
VehicleInterfaces.Interfaces.TransmissionBus.....	40
VehicleInterfaces.Interfaces.TransmissionControlBus.....	40
VehicleInterfaces.Interfaces.ShiftConnector.....	40
VehicleInterfaces.Interfaces.ShiftInput.....	41
VehicleInterfaces.Interfaces.ShiftOutput.....	41
VehicleInterfaces.Mechanics.....	41
VehicleInterfaces.Mechanics.NormalisedTranslational.....	42
VehicleInterfaces.Mechanics.NormalisedTranslational.Interfaces.....	42
VehicleInterfaces.Mechanics.NormalisedTranslational.Interfaces.Flange.....	42
VehicleInterfaces.Mechanics.NormalisedTranslational.Position.....	43
VehicleInterfaces.Mechanics.NormalisedTranslational.Force.....	43
VehicleInterfaces.Mechanics.NormalisedTranslational.PositionSensor.....	43
VehicleInterfaces.Mechanics.NormalisedRotational.....	44
VehicleInterfaces.Mechanics.NormalisedRotational.Interfaces.....	44
VehicleInterfaces.Mechanics.NormalisedRotational.Interfaces.Flange.....	44
VehicleInterfaces.Mechanics.NormalisedRotational.Position.....	45
VehicleInterfaces.Mechanics.NormalisedRotational.Torque.....	45
VehicleInterfaces.Mechanics.NormalisedRotational.AngleSensor.....	45
VehicleInterfaces.Mechanics.MultiBody.....	46
VehicleInterfaces.Mechanics.MultiBody.MultiBodyEnd.....	46
VehicleInterfaces.Types.....	46
VehicleInterfaces.Types.NormalizedReal.....	47
VehicleInterfaces.Types.Gear.....	47
VehicleInterfaces.Types.GearMode.....	47
VehicleInterfaces.Types.GearMode.Temp.....	48
VehicleInterfaces.Types.IgnitionSetting.....	48
VehicleInterfaces.Types.IgnitionSetting.Temp.....	48
VehicleInterfaces.Accessories.....	49
VehicleInterfaces.Accessories.Tutorial.....	49
VehicleInterfaces.Accessories.Interfaces.....	51
VehicleInterfaces.Accessories.Interfaces.Base.....	52
VehicleInterfaces.Accessories.NoAccessories.....	52
VehicleInterfaces.Accessories.MinimalAccessories.....	53
VehicleInterfaces.Atmospheres.....	53
VehicleInterfaces.Atmospheres.Tutorial.....	54
VehicleInterfaces.Atmospheres.Interfaces.....	55
VehicleInterfaces.Atmospheres.Interfaces.windVelocityBase.....	55
VehicleInterfaces.Atmospheres.Interfaces.densityBase.....	56

VehicleInterfaces.Atmospheres.Interfaces.temperatureBase.....	56
VehicleInterfaces.Atmospheres.Interfaces.humidityBase.....	56
VehicleInterfaces.Atmospheres.Interfaces.Base.....	57
VehicleInterfaces.Atmospheres.ConstantAtmosphere.....	57
VehicleInterfaces.Brakes.....	58
VehicleInterfaces.Brakes.Tutorial.....	58
VehicleInterfaces.Brakes.Interfaces.....	64
VehicleInterfaces.Brakes.Interfaces.Base.....	64
VehicleInterfaces.Brakes.Interfaces.TwoAxleBase.....	64
VehicleInterfaces.Brakes.Interfaces.ThreeAxleBase.....	65
VehicleInterfaces.Brakes.Interfaces.FourAxleBase.....	65
VehicleInterfaces.Brakes.Interfaces.StandardBus.....	66
VehicleInterfaces.Brakes.NoBrakes.....	67
VehicleInterfaces.Brakes.MinimalBrakes.....	67
VehicleInterfaces.Brakes.MinimalBrakesUsingPedal.....	68
VehicleInterfaces.Chassis.....	68
VehicleInterfaces.Chassis.Tutorial.....	69
VehicleInterfaces.Chassis.Interfaces.....	76
VehicleInterfaces.Chassis.Interfaces.Base.....	77
VehicleInterfaces.Chassis.Interfaces.TwoAxleBase.....	77
VehicleInterfaces.Chassis.Interfaces.ThreeAxleBase.....	78
VehicleInterfaces.Chassis.Interfaces.FourAxleBase.....	78
VehicleInterfaces.Chassis.Interfaces.StandardBus.....	79
VehicleInterfaces.Chassis.NoChassis.....	79
VehicleInterfaces.Chassis.MinimalChassis.....	80
VehicleInterfaces.Chassis.MinimalChassis2.....	80
VehicleInterfaces.Chassis.MinimalChassis3.....	81
VehicleInterfaces.Controllers.....	82
VehicleInterfaces.Controllers.Tutorial.....	82
VehicleInterfaces.Controllers.Interfaces.....	86
VehicleInterfaces.Controllers.Interfaces.Base.....	86
VehicleInterfaces.Controllers.NoController.....	87
VehicleInterfaces.DataDictionaries.....	87
VehicleInterfaces.DataDictionaries.Tutorial.....	88
VehicleInterfaces.DataDictionaries.Interfaces.....	89
VehicleInterfaces.DataDictionaries.Interfaces.Base.....	89
VehicleInterfaces.DataDictionaries.NoDataDictionary.....	90
VehicleInterfaces.DataDictionaries.MinimalExample.....	90
VehicleInterfaces.Drivelines.....	90
VehicleInterfaces.Drivelines.Tutorial.....	91
VehicleInterfaces.Drivelines.Interfaces.....	95
VehicleInterfaces.Drivelines.Interfaces.Base.....	95
VehicleInterfaces.Drivelines.Interfaces.TwoAxleBase.....	96
VehicleInterfaces.Drivelines.Interfaces.ThreeAxleBase.....	96
VehicleInterfaces.Drivelines.Interfaces.FourAxleBase.....	97
VehicleInterfaces.Drivelines.NoDriveline.....	97
VehicleInterfaces.Drivelines.MinimalDriveline.....	98
VehicleInterfaces.DriverEnvironments.....	99
VehicleInterfaces.DriverEnvironments.Tutorial.....	100
VehicleInterfaces.DriverEnvironments.Interfaces.....	101
VehicleInterfaces.DriverEnvironments.Interfaces.Base.....	102
VehicleInterfaces.DriverEnvironments.Interfaces.BaseAutomaticTransmission.....	102
VehicleInterfaces.DriverEnvironments.Interfaces.BaseAutomaticTransmissionExternalDriver.....	102
VehicleInterfaces.DriverEnvironments.Interfaces.BaseManualTransmission.....	103
VehicleInterfaces.DriverEnvironments.Interfaces.BaseManualTransmissionExternalDriver.....	103
VehicleInterfaces.DriverEnvironments.Interfaces.MinimalBus.....	104
VehicleInterfaces.DriverEnvironments.Interfaces.BusForAutomaticTransmission.....	104
VehicleInterfaces.DriverEnvironments.Interfaces.BusForManualTransmission.....	105

VehicleInterfaces.DriverEnvironments.NoDriverEnvironment.....	105
VehicleInterfaces.DriverEnvironments.DriveByWireAutomatic.....	106
VehicleInterfaces.DriverEnvironments.ConventionalManual.....	107
VehicleInterfaces.DriverEnvironments.DriveByWireAutomaticExternalDriver.....	107
VehicleInterfaces.Drivers.....	108
VehicleInterfaces.Drivers.Tutorial.....	108
VehicleInterfaces.Drivers.Interfaces.....	109
VehicleInterfaces.Drivers.Interfaces.Base.....	110
VehicleInterfaces.Drivers.Interfaces.MinimalBus.....	110
VehicleInterfaces.Drivers.Interfaces.BusForAutomaticTransmission.....	111
VehicleInterfaces.Drivers.Interfaces.BusForManualTransmission.....	111
VehicleInterfaces.Drivers.NoDriver.....	112
VehicleInterfaces.Drivers.MinimalDriver.....	112
VehicleInterfaces.ElectricDrives.....	113
VehicleInterfaces.ElectricDrives.Tutorial.....	113
VehicleInterfaces.ElectricDrives.Interfaces.....	116
VehicleInterfaces.ElectricDrives.Interfaces.Base.....	116
VehicleInterfaces.ElectricDrives.Interfaces.BaseDCMachine.....	117
VehicleInterfaces.ElectricDrives.SimpleMotorDC.....	117
VehicleInterfaces.EnergyStorage.....	118
VehicleInterfaces.EnergyStorage.Tutorial.....	118
VehicleInterfaces.EnergyStorage.Interfaces.....	120
VehicleInterfaces.EnergyStorage.Interfaces.Base.....	120
VehicleInterfaces.EnergyStorage.Battery.....	121
VehicleInterfaces.Engines.....	121
VehicleInterfaces.Engines.Tutorial.....	122
VehicleInterfaces.Engines.Interfaces.....	128
VehicleInterfaces.Engines.Interfaces.Base.....	128
VehicleInterfaces.Engines.Interfaces.StandardBus.....	129
VehicleInterfaces.Engines.NullEngine.....	129
VehicleInterfaces.Engines.MinimalEngine.....	130
VehicleInterfaces.Engines.MinimalEngineUsingPedal.....	131
VehicleInterfaces.PowertrainMounts.....	131
VehicleInterfaces.PowertrainMounts.Tutorial.....	132
VehicleInterfaces.PowertrainMounts.Interfaces.....	136
VehicleInterfaces.PowertrainMounts.Interfaces.Base.....	137
VehicleInterfaces.PowertrainMounts.Interfaces.SingleSystemMount.....	137
VehicleInterfaces.PowertrainMounts.Interfaces.DualSystemMount.....	137
VehicleInterfaces.PowertrainMounts.Interfaces.TripleSystemMount.....	137
VehicleInterfaces.PowertrainMounts.ThreeSystemRigidMount.....	138
VehicleInterfaces.PowertrainMounts.TwoSystemRigidMount.....	138
VehicleInterfaces.PowertrainMounts.SingleSystemRigidMount.....	139
VehicleInterfaces.Roads.....	139
VehicleInterfaces.Roads.Tutorial.....	140
VehicleInterfaces.Roads.Interfaces.....	142
VehicleInterfaces.Roads.Interfaces.positionBase.....	142
VehicleInterfaces.Roads.Interfaces.trackOffsetBase.....	142
VehicleInterfaces.Roads.Interfaces.normalBase.....	143
VehicleInterfaces.Roads.Interfaces.headingDirectionBase.....	143
VehicleInterfaces.Roads.Interfaces.frictionCoefficientBase.....	144
VehicleInterfaces.Roads.Interfaces.Base.....	144
VehicleInterfaces.Roads.FlatRoad.....	144
VehicleInterfaces.Roads.CircleRoad.....	145
VehicleInterfaces.Transmissions.....	146
VehicleInterfaces.Transmissions.Tutorial.....	147
VehicleInterfaces.Transmissions.Interfaces.....	151
VehicleInterfaces.Transmissions.Interfaces.Base.....	152
VehicleInterfaces.Transmissions.Interfaces.BaseAutomaticTransmission.....	152

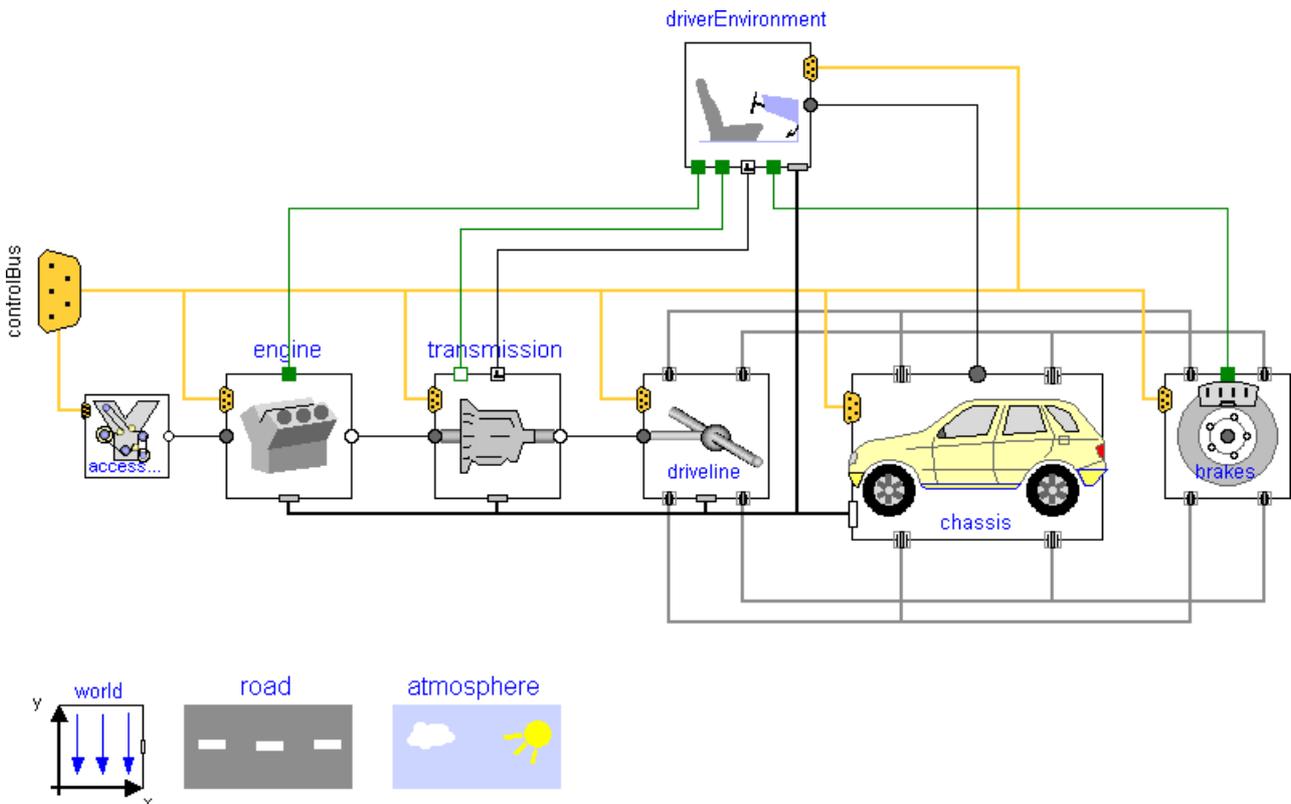
VehicleInterfaces.Transmissions.Interfaces.BaseManualTransmission.....	153
VehicleInterfaces.Transmissions.Interfaces.BaseTwoInputTransmission.....	153
VehicleInterfaces.Transmissions.Interfaces.StandardBus.....	154
VehicleInterfaces.Transmissions.Interfaces.StandardControlBus.....	154
VehicleInterfaces.Transmissions.NoTransmissions.....	155
VehicleInterfaces.Transmissions.SingleGearAutomaticTransmission.....	155
VehicleInterfaces.Transmissions.SingleGearManualTransmission.....	156
VehicleInterfaces.Transmissions.PowerSplitDevice.....	156

VehicleInterfaces

VehicleInterfaces Library (Version 1.2) - Interface definitions and architectures for vehicle system modeling

Information

This Modelica library provides standard interface definitions for automotive subsystems and vehicle models. These are designed to promote compatibility between the various automotive libraries and provide a flexible, powerful structure for vehicle modelling. The main focus of the library is on defining the interfaces for the individual subsystems and a number of vehicle model examples are included to demonstrate how they might be used.



For an automotive library to be compatible with other libraries based on this set of interface definitions they should extend the interface definition from within this library and following the naming convention for the signal bus.

This library also contains a proposal for a naming convention covering the whole model library and the developers of this library would recommend that this convention is followed by other developers to ensure a consistent naming convention across the whole set of automotive models.

The complete documentation of the VehicleInterfaces library in pdf format is available as [VehicleInterfaces.pdf](#)

The library has been developed as a co-operation between a number of vendors who are currently developing automotive libraries. The developers can be contacted by emailing vi@claytex.com, please see [acknowledgements](#) for a list of the people involved in the development of this library.

Copyright © 2005-2011, Dassault Systèmes, DLR and Modelon

Extends from Modelica.Icons.Package (Icon for standard packages).

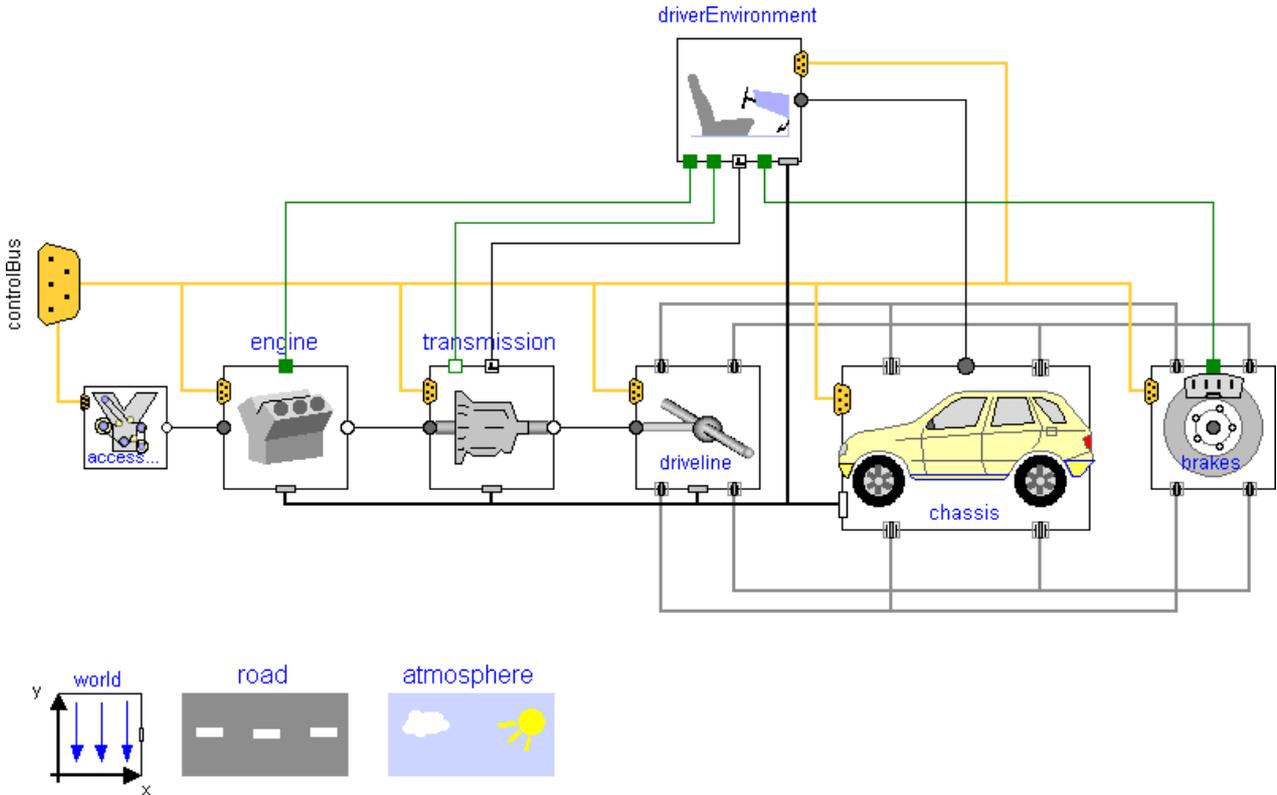
Package Content

Name	Description
 UsersGuide	Users Guide
 Examples	Collection of vehicle model examples
 Blocks	Collection of input/output fixed causality blocks
 Icons	Collection of icons
 Interfaces	Collection of general interface definitions
 Mechanics	Collection of additional mechanical components
 Types	Collection of type definitions
 Accessories	Collection of accessory subsystem definitions
 Atmospheres	Collection of atmosphere subsystem definitions
 Brakes	Collection of brakes subsystem definitions
 Chassis	Collection of chassis subsystem definitions
 Controllers	Collection of controllers subsystem definitions
 DataDictionaries	Collection of data dictionary definitions
 Drivelines	Collection of drivelines subsystem definitions
 DriverEnvironments	Collection of driver environment subsystem definitions
 Drivers	Collection of driver subsystem definitions
 ElectricDrives	Collection of electric drive subsystem definitions
 EnergyStorage	Collection of energy storage subsystem definitions
 Engines	Collection of combustion engine subsystem definitions
 PowertrainMounts	Collection of powertrain mounts subsystem definition
 Roads	Collection of road definitions
 Transmissions	Collection of transmission subsystem definitions

VehicleInterfaces.UsersGuide

The VehicleInterfaces Modelica library provides standard interface definitions for automotive subsystems and vehicle models. These are designed to promote compatibility between the various automotive libraries and provide a flexible, powerful structure for vehicle modelling. The main focus of the library is on defining the interfaces for the individual subsystems and a number of vehicle model examples are included to demonstrate how they might be used.





For an automotive library to be compatible with other libraries based on this set of interface definitions they should extend the interface definition from within this library and following the naming convention for the control signal bus and driver interaction bus if used.

This library also contains a proposal for a naming convention covering the whole model library and the developers of this library would recommend that this convention is followed by other developers to ensure a consistent naming convention across the whole set of automotive models.

The library has been developed as a co-operation between a number of vendors who are currently developing automotive libraries. The developers can be contacted by emailing vi@claytex.com, please see [here](#) for a list of the people involved in the development of this library.

Package Content

Name	Description
 QuickStart	Quick start guide
 NamingConventions	Naming conventions
 SignalBus	Control signal bus
 DriverInteractionBus	Driver interaction bus
 ModelRotatingParts	Modelling rotating systems
 SubsystemDefinitions	Subsystem descriptions
 Tutorials	Tutorials
 ReleaseNotes	Release notes
 Acknowledgements	Acknowledgements
 License	VehicleInterfaces license

VehicleInterfaces.UsersGuide.QuickStart

This section will quickly highlight some of the features of the library.



Getting started

A new subsystem should extend a class from within `VehicleInterfaces.SubsystemTypeName.Interfaces`. These classes only provide the basic connection interface required. If you also want to use the standard Icon for your new subsystem these can be found in `VehicleInterfaces.Icons`.

Examples

A number of vehicle model examples are included for four wheeled vehicles and could be duplicated or extended to create your own vehicle models. All the subsystems are defined as replaceable so they can be changed to the required definition after extending from the example. Control system blocks can be added as required by dragging the controller model into the vehicle model and connecting the controller to the control bus.

WheelHub connectors

The WheelHub connectors are based on a new connector found in `Modelica.Mechanics.MultiBody.Interfaces` called `FlangeWithBearing`. This connector contains a 1D rotational connector and an optional MultiBody connector. The MultiBody connector can be enabled/disabled via a parameter in the connector definition. In the case of the VehicleInterfaces subsystem definitions these parameters are propagated to the subsystem level so that they can all be set by one parameter. The MultiBody connector is used to provide the support for the rotating component.

For example, the MultiBody connector in the WheelHub connection between the driveline and chassis subsystems provides a way to support the ends of the halfshafts when modelling the driveline as a MultiBody system. In the driveline and chassis subsystem templates this MultiBody connector is enabled via a parameter called **includeWheelBearings**. Both subsystems must have the same setting for this parameter for them to be compatible. Dymola generates a warning about unmatched connectors if this is not the case. To access the parameters, first extend the template to create your new driveline model. Then go to the component browser and right click on the extended class and select parameters. The parameter dialog will now be displayed and allow you access to the connector parameters.

Optional connectors

Many of the subsystem templates contain optional connectors. These are highlighted in the template diagrams by an orange dashed line around the connector. These connectors are enabled by parameters.

For example: The driveline subsystem contains an optional MultiBody connector called **drivelineMount**. This is enabled via a parameter called **includeMount**. To access the parameters, first extend the template to create your new driveline model. Then go to the component browser and right click on the extended class and select parameters. The parameter dialog will now be displayed and allow you access to the connector parameters.

Control Bus Structure

The control bus structure utilises expandable connectors to provide a flexible bus concept. This approach makes it easy to add new signals to the control bus so that data can be passed around the model. The control bus structure implemented is hierarchical so that signals are easily located. The approach adopted is that physical measurements taken from, for example, the transmission are placed on a sub-bus called **transmissionBus**. Signals output from control blocks, for example the transmission controller, are placed on a sub-bus called **transmissionControlBus**.

VehicleInterfaces.UsersGuide.NamingConventions



This section describes the naming conventions used in the VehicleInterfaces Library. The naming convention used throughout this library is also recommended as the naming convention to be used within compatible Automotive libraries but the user should refer to the documentation of the libraries they are using to confirm that this naming convention is used.

The convention on the naming of bus signals is the most important aspect of the convention that needs to be followed to ensure compatibility between libraries.

Naming of classes and components

Classes (for packages, models, records, etc.) are written with initial capital letters for each word, like in "DriverEnvironments".

Instances are written with a lower-case initial letter and then initial capital letters for each word, like in "driverEnvironment".

Variables and parameters that are derived from atomic types (Boolean, Integer, Real, String) names are written with a lower-case initial letter and then initial capital letters for each word, like in "engineSpeed". Abbreviated variable/parameter names are written with lower case letters, unless there is a strong convention, e.g. temperature is normally written "T".

Indices are denoted with an underscore, like in "r_x".

Naming of Variables and Parameters

Class instances normally have names that refer to the function/position of the instances, rather than the class name, like in "leftLinkage" for an instance of a "TrailingArm" linkage.

Parameters and variables have descriptive names, like in "engineSpeed", when it is practical. Short (single-letter) names or abbreviations are used when that is considered more natural, and are then named after the property they describe, like in "v_x". For short names, indices are separated by underscore.

Initial Values

Parameters that represent initial values for simulation experiments are denoted with the index **start**, like "v_start".

Naming of Properties

Physical properties are named according to the standards established in the Modelica Standard Library:

r	Position
v	Velocity
a	Acceleration
phi	Rotation angle
w	Angular velocity
z	Angular acceleration
f	Force
tau	Torque
n	Direction of rotation or translation
m	Mass
c	Stiffness
d	Damping
J	Inertia element, (gear) ratio
k	Amplification/gain

s Distance

Wheel Identifiers

Wheels are identified by numbers from front left towards rear right. Those are, typically, used as indices in parameter or variable names. E.g., for a four-wheel car the front left wheel is "1", front right is "2", rear left "3", and rear right "4".

VehicleInterfaces.UsersGuide.SignalBus



Every subsystem has access to the control signal bus which is modelled using expandable connectors. This enables signals to be easily added to the control signal bus to suit a particular application. A minimal set of signals has been defined to ensure basic compatibility between the various automotive libraries currently being developed.

The control signal bus should not contain physical connectors but should contain sensed or actuation values as signal connections.

Naming of bus signals

Signals placed on the control signal bus should be named following the same rules as naming the instance of a class. This is based on the view that the items placed on to the control signal bus are actually connectors rather than simple variable values.

Signals placed on the bus are written with a lower-case initial letter and then initial capital letters for each word, like in "driverEnvironment".

The names of the connectors placed on to the bus should start with the name of the subsystem that is outputting the connector on to the bus. This should then be followed by "Bus." and then a descriptive name for the connector, like "engineBus.engineSpeed" or "chassisBus.longitudinalVelocity"

Minimal signal set

The signals in the following table are common to all vehicle types and should be combined with the additional signals defined for the different vehicle variants (i.e. manual and automatic transmission equipped vehicles).

Signal name	Units	Signal description
brakesBus.wheelSpeed_n	rad/s	Individual wheel speed where n is an integer describing which wheel this is from.
chassisBus.longitudinalVelocity	m/s	Vehicle Longitudinal Velocity
driverBus.ignition	Enumeration	Signals whether the driver wants the vehicle to be operating. There are 3 possible values: Off, On or Start (VehicleInterfaces.Types.IgnitionSetting)
engineBus.speed	rad/s	Engine speed
transmissionBus.outputSpeed	rad/s	Transmission output shaft speed
transmissionControlBus.currentGear	Integer	Currently selected gear

The following additional signals are required for **manual** transmission vehicles.

Signal name	Units	Signal description
transmissionBus.clutchLocked	Boolean	True if the clutch is locked, otherwise false

The following additional signals are required for an **automatic** transmission vehicles.

Signal name	Units	Signal description
-------------	-------	--------------------

driverBus.gearboxMode	Enumeration	Gearbox Mode (enumeration when supported, currently uses Integers defined according to VehicleInterfaces.Types.GearMode)
driverBus.requestedGear	Integer	Driver requested gear for use in manual or limited gearbox modes

Additional signals for a drive-by-wire vehicle

The following additional signals are required to model a drive-by-wire vehicle and are common for both manual and automatic transmission variants.

Signal name	Units	Signal description
driverBus.acceleratorPedalPosition	None	Driver accelerator pedal position - normalised Real, 0 = pedal released, 1 = pedal fully pressed
driverBus.brakePedalPosition	None	Driver brake pedal position - normalised Real, 0 = pedal released, 1 = pedal fully pressed

The following additional signals are required for a **manual** transmission drive-by-wire vehicle.

Signal name	Units	Signal description
driverBus.clutchPedalPosition	None	Driver clutch pedal position - normalised Real, 0 = pedal released, 1 = pedal fully pressed
driverBus.gear	None	Driver selected gear

Package Content

Name	Description
 AddingSignals	Adding signals to the bus

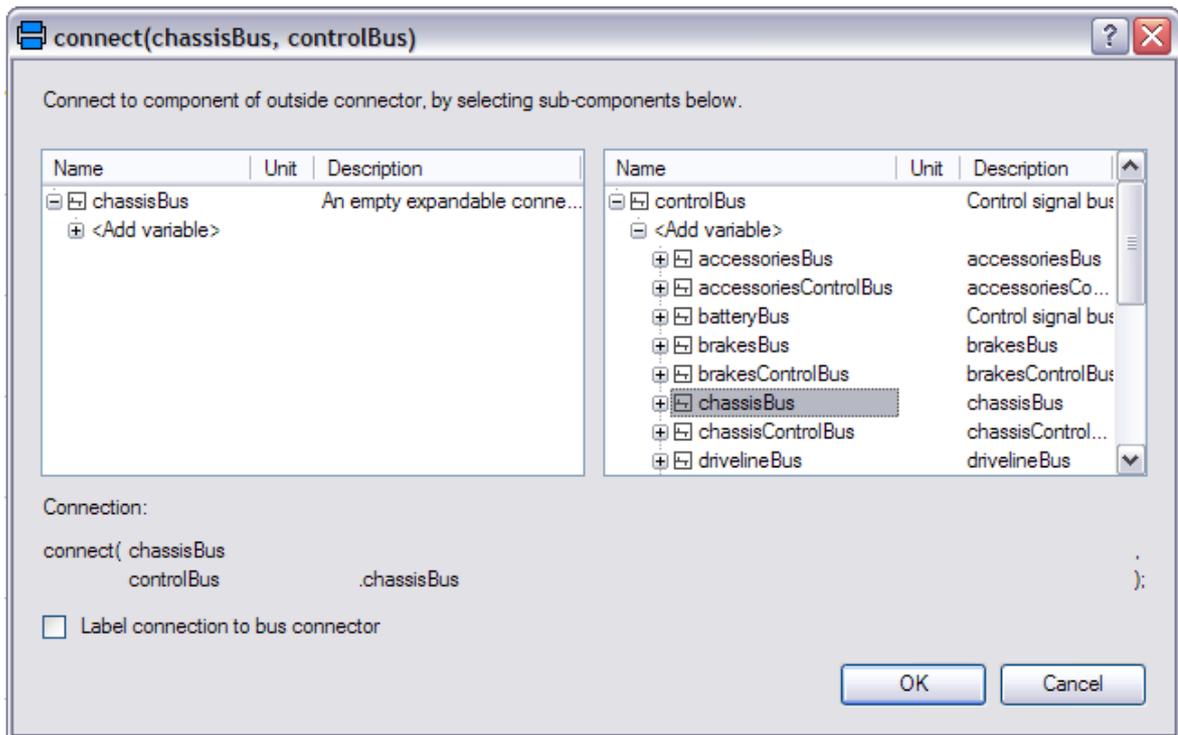
VehicleInterfaces.UsersGuide.SignalBus.AddingSignals

This method demonstrates how to add signals to the control signal bus. To place an existing signal such as vehicle longitudinal velocity on to the signal bus in the chassis subsystem use the following process:

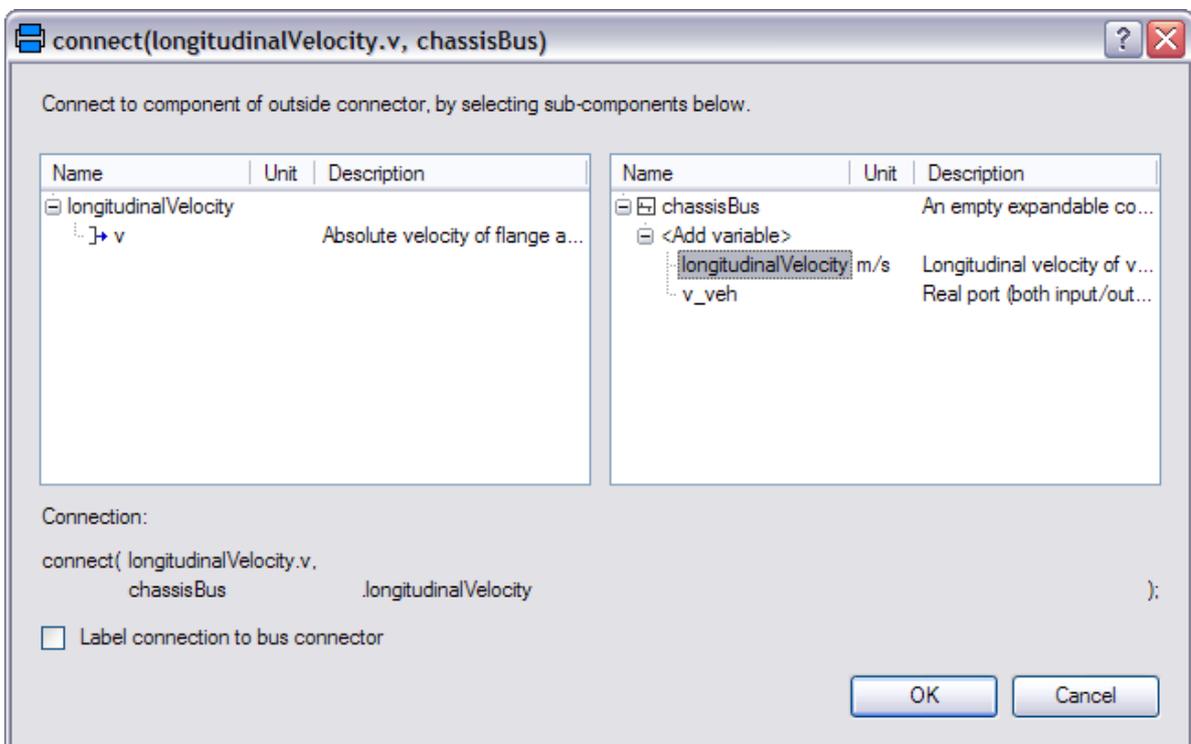


Working within your chassis model...

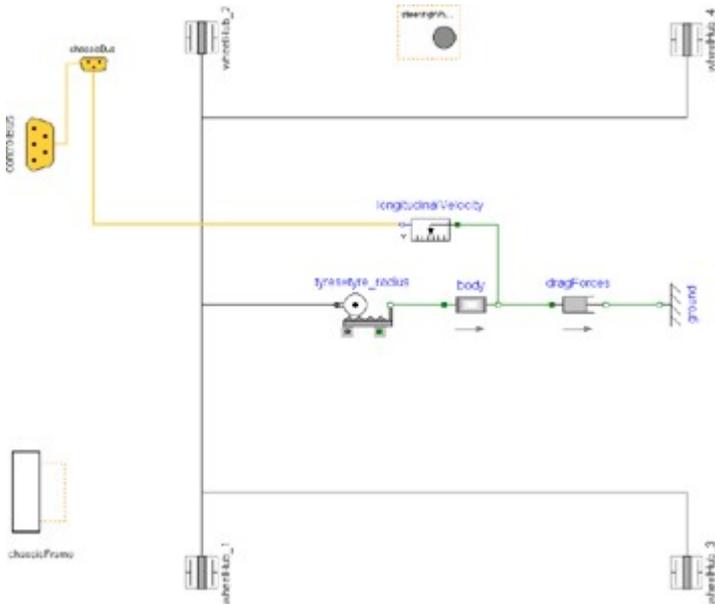
1. If not already there, drag the **VehicleInterfaces.Interface.ChassisBus** connector in to the model. It should get the default name **chassisBus** but rename it to this if it doesn't.
2. Draw a connection from this new connector to the controlBus connector.
3. In the connection dialog generated by Dymola select the following name: **chassisBus**



4. Add a sensor to measure the vehicle speed and drag a connection to the **chassisBus** connector just added.
5. In the connection dialog generated by Dymola select **longitudinalVelocity** from the list of options.



The chassis subsystem should look something like:



There are two methods for adding a new signal to the controlBus.

1. You draw the connection as in step 4 above, and instead of selecting an option from the list in step 5 you would select the "Add variable" option and then type the name of the signal you would like to add to the controlBus.
2. You extend from the chassisBus connector definition that can be found in **VehicleInterfaces.Chassis.Internal.StandardBus** and add the new signal definition. Then, when you reach steps 4 and 5 (above) your new signal will appear in the list of options and can be selected.

VehicleInterfaces.UsersGuide.DriverInteractionBus



The Driver Interaction bus connects the DriverEnvironment to an external Driver model and is implemented using expandable connectors. This enables signals to be easily added to the control signal bus to suit a particular application. A minimal set of signals has been defined to ensure basic compatibility between the various automotive libraries currently being developed.

The Driver Interaction bus can contain both normalised physical connectors and sensed values as signal connections.

Naming of Driver Interaction bus components

Connectors placed on the Driver Interaction bus should be named following the same rules as naming the instance of a class. This is based on the view that the items placed on to the control signal bus are actually connectors rather than simple variable values.

This means that connectors placed on the bus are written with a lower-case initial letter and then initial capital letters for each word, like in "acceleratorPedal".

Standard names

The following table defines the standard names and associated connector types and units used.

Name on the bus	Connector Type	Units	Description
acceleratorPedal	NTI.Flange	-	Accelerator pedal connection using a normalised physical connector
brakePedal	NTI.Flange	-	Brake pedal connection using a normalised physical

			connector
clutchPedal	NTI.Flange	-	Clutch pedal connection using a normalised physical connector
steeringWheel	NRI.Flange_a	-	Steering wheel connection using a normalised physical connector
vehicleSpeed	RealSignal	m/s	Vehicle longitudinal speed in kmh
engineSpeed	RealSignal	rad/s	Engine speed in rpm
gear	IntegerSignal	-	Gear selected by the driver (manual transmission only)
clutchLocked	BooleanSignal	-	Flag for clutch state, true=clutch in stuck mode (manual transmission only)
gearboxMode	IntegerSignal	Enumeration	Gearbox mode selected by the driver (automatic transmission only)
requestedGear	IntegerSignal	-	Gear selected by the driver if the automatic is in a manual mode (automatic transmission only)

Key:

NTI = NormalisedTranslational.Interfaces

NRI = NormalisedRotational.Interfaces

RealSignal = Modelica.Blocks.Interfaces.RealSignal

IntegerSignal = Modelica.Blocks.Interfaces.IntegerSignal

BooleanSignal = Modelica.Blocks.Interfaces.BooleanSignal

VI = VehicleInterfaces

Pedal Conventions

For the driver the following conventions are used to describe the pedal positions as normalised values.

- 1 = Pedal Fully Pressed
- 0 = Pedal Fully Released

For example: when the accelerator pedal position = 1, the driver is accelerating as fast as possible; when the clutch pedal position = 1, the driver has pressed the clutch pedal and the clutch is actually disengaged.

VehicleInterfaces.UsersGuide.ModelRotatingParts

Within VehicleInterfaces rather than limiting our-selves to modelling rotating components as a simple 1D rotation we have built in the flexibility to model rotating components as MultiBody systems. This has been possible through the development of a new connector called FlangeWithBearing. This is a hierarchical connector that contains a 1D rotational connector and a conditional MultiBody connector. The use of the MultiBody connector is controlled by a parameter in the connector. This enables the FlangeWithBearing connector to model rotational effects as a purely 1D system or it can correctly include MultiBody effects.



Where this connector is used in an interface definition the parameter includeBearingConnector is linked to a parameter at the model level so that the model developer can easily activate this connector if required. These parameters are declared as protected parameters so that they are only available to the model developer as they create the subsystem model and cannot be changed when the model is used.

The use of the bearing connectors needs to be carefully considered to avoid inadvertently creating mechanical loops in the model. The following example guidelines for the engine and transmission subsystems should help highlight the issues so that the model developer can determine when it is appropriate to use the bearing connectors. For the engine and transmission subsystem models:

1. When they are modelled as a pure 1D rotational system then no bearing connectors are required.
2. When they are modelled as a 1D rotational system with reactions on to a MultiBody system then no bearing connectors are required.
3. When they are being modelled as a MultiBody system but they are rigidly connected together then

the bearing frame between the engine and transmission should not be included. In this case the transmissionMount connector should support the MultiBody elements of the transmission. The rest of the model then needs to be considered before deciding whether to include the bearing between the transmission and driveline or between the engine and accessories subsystems.

4. When they are being modelled as a MultiBody system but they are not rigidly connected together then the bearing frame between the engine and transmission will be required to support the intermediate drive shaft.

VehicleInterfaces.UsersGuide.SubsystemDefinitions

The library contains the following subsystem definitions:

- [Accessories](#) - engine driven accessories
- [Atmospheres](#) - ambient conditions
- [Brakes](#) - brake system
- [Chassis](#) - chassis system
- [Controllers](#) - generic controller interface
- [DataDictionaries](#) - for signal aliasing
- [Drivelines](#) - driveline system
- [DriverEnvironments](#) - driver vehicle interface
- [Drivers](#) - optional driver subsystem
- [ElectricDrives](#) - electric machine interface
- [Engines](#) - engine interface
- [PowertrainMounts](#) - powertrain mounting systems
- [Roads](#) - road definition
- [Transmissions](#) - transmission subsystem



VehicleInterfaces.UsersGuide.Tutorials

Tutorials are provided as part of the VehicleInterfaces Users Guide and these demonstrate:

1. [How to create a vehicle model using one of the example architectures](#)
2. Each subsystem also contains a Tutorial class that demonstrates the use of that subsystem



A tutorial session on the VehicleInterfaces package was run at the Modelica 2006 conference. The tutorial material from this conference can be found at <http://www.modelica.org/events/modelica2006/Proceedings/html/tutorials.html>. Some elements of the library may have been renamed since this tutorial but the principles are still utilised.

Package Content

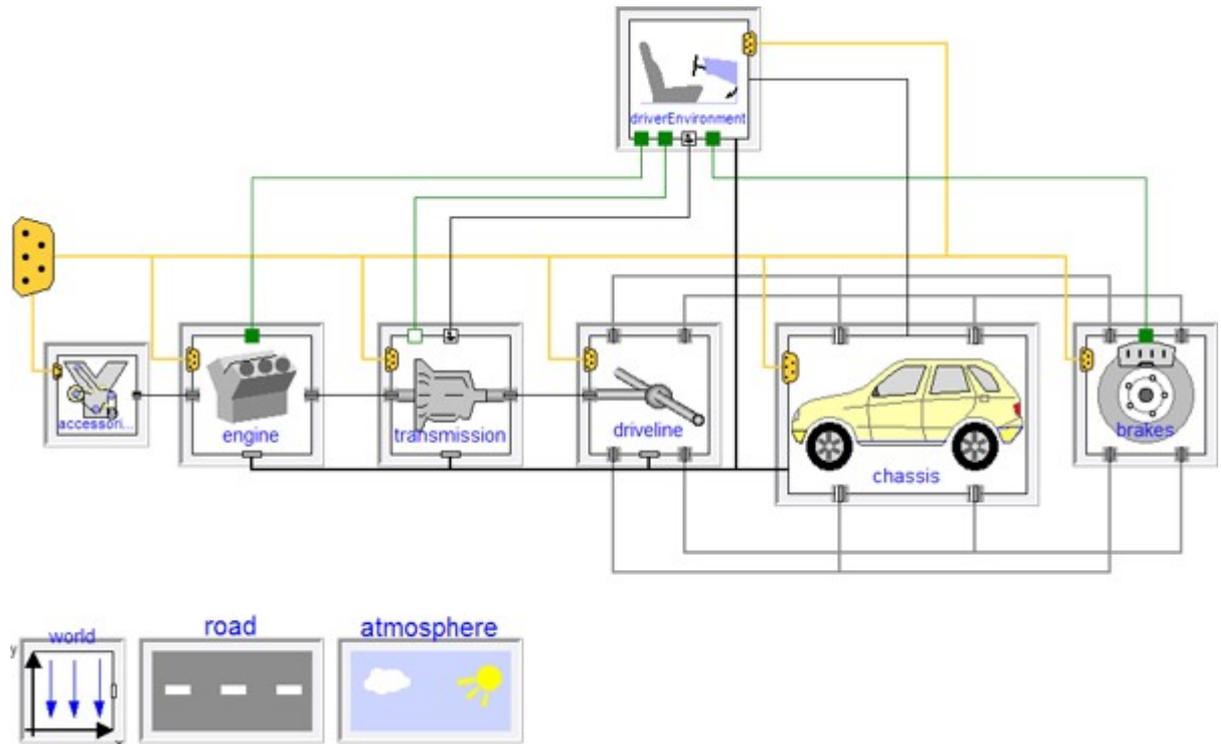
Name	Description
 Tutorial1	Creating a vehicle model by extending from an Example

VehicleInterfaces.UsersGuide.Tutorials.Tutorial1

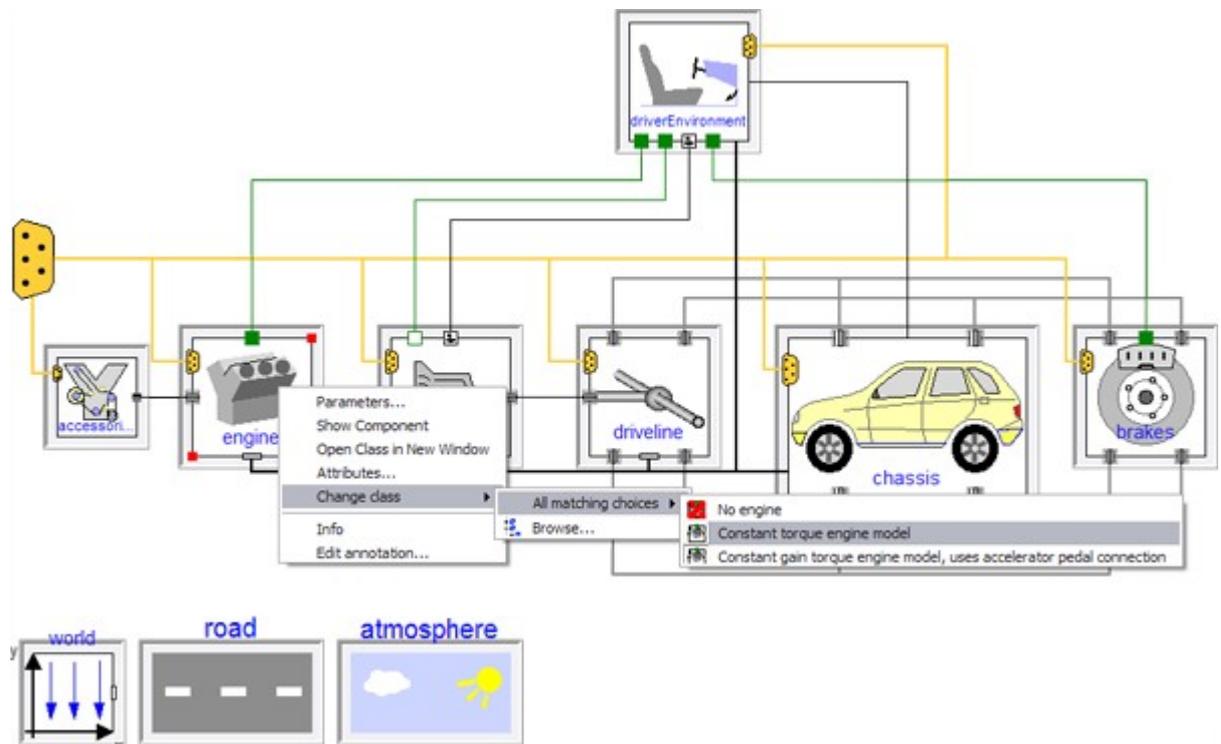
The following process will demonstrate how to create a new vehicle model using one of the example model architectures as a template. This tutorial will guide you through building a conventional passenger car, i.e. a vehicle with 4 wheels.

1. Create a new model that extends **VehicleInterfaces.Exmaples.ConventionalManualVehicle**, it should look like this:





- By right clicking on each subsystem in turn you can now select the appropriate models for each subsystem.
Using Dymola, the context menu should look like below when changing the engine subsystem



- When you have set the appropriate model for each subsystem your model is ready for parametrisation and simulation.

VehicleInterfaces.UsersGuide.ReleaseNotes

This section summarizes the changes that have been performed on the library.
Extends from Modelica.Icons.ReleaseNotes (Icon for release notes in documentation).



Package Content

Name	Description
 Version_1_2	Version 1.2 (August 8, 2011)
 Version_1_1_2	Version 1.1.2 (Nov. 8, 2010)
 Version_1_1_1	Version 1.1.1 (Jun. 30, 2009)
 Version_1_1	Version 1.1 (Feb. 5, 2008)
 Version_1_0	Version 1.0 (Dec. 3, 2007)

VehicleInterfaces.UsersGuide.ReleaseNotes.Version_1_2

This version requires Dymola 2012 or higher and the Modelica 3.2 Library. This version is backwards compatible to versions 1.1.x of the VehicleInterfaces library.



Improvements in this version:

- **Improved buses**
In order to simplify working with buses, all buses in VehicleInterfaces are defined as replaceable expandable empty connectors. Whenever the buses are used concretely, they are redeclared to a connector model where all signal definitions are present. As a result, the actual signal definitions (with units and description texts) are always available at the place where they are used. Still, users can add more signals or need not use all defined signals (due to the new feature of Modelica 3.2, that not used signals of an expandable connector are ignored).
- **Adapted to the conventions of the Modelica Standard Library 3.2**
(directory "VehicleInterfaces/Extras" moved to the new directory "VehicleInterfaces/Resources"; all links to images, scripts, documentation etc. have been changed to URIs such as modelica://VehicleInterfaces/Resources/Images/xx.png; icon layout adapted to MSL 3.2 layout).
- VehicleInterfaces.**EnergyStorage**.Interfaces.Base:
A optional ground and an optional heat port have been added. Both components can be enabled via a corresponding parameter

Extends from Modelica.Icons.ReleaseNotes (Icon for release notes in documentation).

VehicleInterfaces.UsersGuide.ReleaseNotes.Version_1_1_2

Adapted to Modelica Standard Library 3.2. This version is backward compatible to the version 1.1.1.

Extends from Modelica.Icons.ReleaseNotes (Icon for release notes in documentation).



VehicleInterfaces.UsersGuide.ReleaseNotes.Version_1_1_1

Further adaption to Modelica Language 3 and Modelica Standard Library 3.1. This version is backward compatible to the version 1.1.

As enabled by Dymola 7, the library check was performed on the VehicleInterfaces library utilizing the ModelManagement Library. Consequently, especially the documentation has been upgraded and fulfilled. Main changes with respect to the version 1.1:



- [VehicleInterfaces.Mechanics](#): Small changes at icon and diagram layer to be Modelica 3 compliant.
- [VehicleInterfaces.Mechanics.MultiBody.MultiBodyEnd](#) adapted, so that "Mixing of components and equations" is avoided.
- [VehicleInterfaces.UsersGuide.NamingConventions](#): changed "a" to "z" for an *angular* acceleration.
- Changed icon for [VehicleInterfaces.Examples](#).
- Added icon [BaseClassPackage](#) to be used for packages with base classes and internal material.

Extends from [Modelica.Icons.ReleaseNotes](#) (Icon for release notes in documentation).

VehicleInterfaces.UsersGuide.ReleaseNotes.Version_1_1

Adapted to Modelica Language 3 and Modelica Standard Library 3.0. This version is slightly non-backwards compatible to the previous version 1.0. A conversion script is provided to automatically convert to the new version. Changes with respect to version 1.0:



- Graphical annotations changed to MSL3.0.
- Changed definition of Road and Atmosphere, since previous definition was not Modelica 3 compliant.
- [VehicleInterfaces.Mechanics.MultiBody.MultiBodyEnd](#) adapted, so that `ObsoleteModelica3` is not used (was introduced by conversion script of Modelica Standard Library 3.0).
- [VehicleInterfaces.Types.Conversions](#) removed, since no longer needed (since tools support `displayUnit`).
- [VehicleInterfaces.Engines.MinimalEngine](#), [VehicleInterfaces.Engines.MinimalEngineUsingPedal](#): Replaced "`NonSI.AngularVelocity_rpm engineSpeed_start`" by `SI.AngularVelocity` with `displayUnit="r/min"` and adapted flywheel Inertia initialization correspondingly.
- [VehicleInterfaces.Roads.Internal.CheckContactCalculation](#) adapted, so that it is according to Modelica 3 (equations for input variables of "tyre" moved from equation section into modifier of "tyre").
- [VehicleInterfaces.Atmospheres.ConstantAtmosphere](#) icon corrected: Automatic Dymola conversion was not corrected, since icon was transformed to 100,100 icon instead of 200,100 icon (complex situation because icon size was actually inherited from [Icons.Atmospheres](#) and it is understandable that the conversion did not recognized this).
- [VehicleInterfaces.Atmospheres.ConstantAtmosphere](#): Removed "`...degC(20)`" function call by "20", since `displayUnit` of temperatures is "degC".
- Removed definition "`import NonSI = xx`" at the top level (is no longer referenced).
- Renamed [VehicleInterfaces.Examples.BaseCar](#) to `PartialVehicle`.
- [VehicleInterfaces.UsersGuide.ReleaseNotes](#) introduced.

Extends from [Modelica.Icons.ReleaseNotes](#) (Icon for release notes in documentation).

VehicleInterfaces.UsersGuide.ReleaseNotes.Version_1_0

First version of [VehicleInterfaces](#) library finalized, based on Modelica Standard Library 2.2.2

Extends from [Modelica.Icons.ReleaseNotes](#) (Icon for release notes in documentation).



VehicleInterfaces.UsersGuide.Acknowledgements

A number of automotive library developers and consultants have co-operated to develop this release of the VehicleInterfaces Library. The developers can all be contacted by emailing vi@claytex.com. The developers are:

- Arsenal Research: Franz Pirker, Anton Haumer, Christian Kral
- Claytex Services Ltd: Mike Dempsey
- Dassault Systèmes AB: Hilding Elmqvist
- DLR Oberpfaffenhofen: Martin Otter, Christian Schweiger, Jakub Tobolar
- DLR Stuttgart: Peter Treffinger, Marcus Baur, Jörg Ungethüm
- Modelon AB: Johan Andreasson, Magnus Gäfvert
- Ricardo UK Ltd: Mark Ingram, Peter Harman

This library has been developed from work on the original Modelica VMA ([Paper from the Modelica Conference 2003](#)) developed by Michael Tiller et al and published by Ford Motor Company.

Additional ideas from intermediate work by members of DLR Oberpfaffenhofen and Modelon has also been incorporated. The above partners agreed to develop these ideas in to a new library and adopt this set of interface definitions.

The implementation of the VehicleInterfaces library was led by Claytex Services Limited under contract to Dassault Systèmes AB



VehicleInterfaces.UsersGuide.License

The VehicleInterfaces library is distributed under the Modelica license (Version 1.1)

Redistribution and use in source and binary forms, with or without modification are permitted, provided that the following conditions are met:

1. The author and copyright notices in the source files, these license conditions and the disclaimer below are (a) retained and (b) reproduced in the documentation provided with the distribution.
2. Modifications of the original source files are allowed, provided that a prominent notice is inserted in each changed file and the accompanying documentation, stating how and when the file was modified, and provided that the conditions under (1) are met.
3. It is not allowed to charge a fee for the original version or a modified version of the software, besides a reasonable fee for distribution and support. Distribution in aggregate with other (possibly commercial) programs as part of a larger (possibly commercial) software distribution is permitted, provided that it is not advertised as a product of your own.

Disclaimer

The software (sources, binaries, etc.) in their original or in a modified form are provided "as is" and the copyright holders assume no responsibility for its contents what so ever. Any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are **disclaimed**. **In no event** shall the copyright holders, or any party who modify and/or redistribute the package, **be liable** for any direct, indirect, incidental, special, exemplary, or consequential damages, arising in any way out of the use of this software, even if advised of the possibility of such damage.

VehicleInterfaces.Examples

Collection of vehicle model examples

Information

Example vehicle model architectures. These can be used as the basis for your own vehicle models but are also fully functional vehicle models on their own.

Extends from Modelica.Icons.ExamplesPackage (Icon for packages containing runnable examples).



Package Content

Name	Description
 ConventionalAutomaticVehicle	Conventional automatic transmission vehicle
 ConventionalVehicle_AltNames	Extension of ConventionalAutomaticVehicle demonstrating the data dictionary
 RearWheelDriveAutomaticVehicle	A conventional front-wheel drive automatic transmission vehicle with separate powertrain mounting system for the front axle and engine-transmission systems
 ConventionalAutomaticVehicleExternalDriver	Conventional automatic transmission vehicle using separate driver and driver-interface components
 ConventionalManualVehicle	Conventional manual transmission vehicle
 FrontWheelDriveManualVehicle	A conventional front-wheel drive manual transmission vehicle including the powertrain mounting systems
 PowerSplitHybrid	Power-split hybrid vehicle model
 SeriesHybrid	Series hybrid vehicle model
 PartialVehicle	Basic model for a four wheeled vehicle, requires powertrain components

VehicleInterfaces.Examples.ConventionalAutomaticVehicle

Conventional automatic transmission vehicle



Information

Example model architecture of a passenger car with an automatic transmission. In this architecture variant the PowerTrain systems are rigidly mounted to the chassis when MultiBody effects are included.

Extends from Modelica.Icons.Example (Icon for runnable examples).

Parameters

Name	Description
Plant Models	
engine	Engine subsystem
transmission	Transmission subsystem
chassis	Chassis subsystem
driveline	Driveline subsystem
brakes	Brakes subsystem
accessories	Accessories subsystem
Driver Models	
driverEnvironment	Driver environment
Conditions	
road	Road model
atmosphere	Atmospheric model
world	Global co-ordinate system

VehicleInterfaces.Examples.ConventionalVehicle_AltNames

Extension of ConventionalAutomaticVehicle demonstrating the data dictionary

**Information**

Example model architecture of a passenger car with an automatic transmission. Based on [ConventionalAutomaticVehicle](#) this example shows how the DataDictionary can be used.

Extends from [ConventionalAutomaticVehicle](#) (Conventional automatic transmission vehicle).

Parameters

Name	Description
Plant Models	
engine	Engine subsystem
transmission	Transmission subsystem
chassis	Chassis subsystem
driveline	Driveline subsystem
brakes	Brakes subsystem
accessories	Accesories subsystem
Driver Models	
driverEnvironment	Driver environment
Conditions	
road	Road model
atmosphere	Atmospheric model
world	Global co-ordinate system
Control Systems	
dataDictionary	Data dictionary

VehicleInterfaces.Examples.RearWheelDriveAutomaticVehicle

A conventional front-wheel drive automatic transmission vehicle with separate powertrain mounting system for the front axle and engine-transmission systems

**Information**

Example model architecture of a rear wheel drive passenger car with an automatic transmission. In this architecture variant the PowerTrain systems can be elastically mounted in the chassis when MultiBody effects are included.

Extends from Modelica.Icons.Example (Icon for runnable examples).

Parameters

Name	Description
Plant Models	
engine	Engine subsystem
transmission	Transmission subsystem
chassis	Chassis subsystem

driveline	Driveline subsystem
brakes	Brakes subsystem
accessories	Accessories subsystem
engineTransmissionMount	Engine and Transmission mounting system
diffMount	Differential mounting system
Driver Models	
driverEnvironment	Driver environment
driver	Driver model
Conditions	
road	Road model
atmosphere	Atmospheric model
world	Global co-ordinate system

VehicleInterfaces.Examples.ConventionalAutomaticVehicleExternalDriver

Conventional automatic transmission vehicle using separate driver and driver-interface components



Information

Example model architecture of a passenger car with an automatic transmission and a driver model that is defined separately from the driverEnvironment. In this architecture variant the PowerTrain systems are rigidly mounted to the chassis when MultiBody effects are included.

Extends from Modelica.Icons.Example (Icon for runnable examples).

Parameters

Name	Description
Plant Models	
engine	Engine subsystem
transmission	Transmission subsystem
chassis	Chassis subsystem
driveline	Driveline subsystem
brakes	Brakes subsystem
accessories	Accessories subsystem
Driver Models	
driverEnvironment	Driver environment
driver	Driver model
Conditions	
road	Road model
atmosphere	Atmospheric model
world	Global co-ordinate system

VehicleInterfaces.Examples.ConventionalManualVehicle

Conventional manual transmission vehicle



Information

Example model architecture of a passenger car with a manual transmission. In this architecture variant the PowerTrain systems are rigidly mounted to the chassis when MultiBody effects are included.

Extends from Modelica.Icons.Example (Icon for runnable examples).

Parameters

Name	Description
Plant Models	
engine	Engine subsystem
transmission	Transmission subsystem
chassis	Chassis subsystem
driveline	Driveline subsystem
brakes	Brakes subsystem
accessories	Accessory subsystem
Driver Models	
driverEnvironment	Driver environment
Conditions	
road	Road model
atmosphere	Atmospheric model
world	Global co-ordinate system

VehicleInterfaces.Examples.FrontWheelDriveManualVehicle

A conventional front-wheel drive manual transmission vehicle including the powertrain mounting systems

**Information**

Example model architecture of a front wheel drive passenger car with an automatic transmission. In this architecture variant the PowerTrain systems can be elastically mounted in the chassis when MultiBody effects are included.

Extends from Modelica.Icons.Example (Icon for runnable examples).

Parameters

Name	Description
Plant Models	
engine	Engine subsystem
transmission	Transmission subsystem
chassis	Chassis subsystem
driveline	Driveline subsystem
brakes	Brakes subsystem
accessories	Accessories subsystem
powertrainMounts	Powertrain mounting system
Driver Models	
driverEnvironment	Driver environment

Conditions	
road	Road models
atmosphere	Atmospheric model
world	Global co-ordinate system

VehicleInterfaces.Examples.PowerSplitHybrid

Power-split hybrid vehicle model



Information

Example model architecture of a passenger car with a power-split hybrid powertrain. In this architecture variant the PowerTrain systems are rigidly mounted to the chassis when MultiBody effects are included.

Extends from [VehicleInterfaces.Examples.PartialVehicle](#) (Basic model for a four wheeled vehicle, requires powertrain components).

Parameters

Name	Description
Plant Models	
chassis	Chassis subsystem
driveline	Driveline subsystem
brakes	Brakes subsystem
engine	Engine subsystem
accessories	Accessories subsystem
motor1	Electric Motor 1
motor2	Electric Motor 2
powerSplitDevice	Power-split device
battery	Energy storage subsystem
Driver Models	
driverEnvironment	Driver environment
Conditions	
road	Road model
atmosphere	Atmospheric model
world	Global co-ordinate system

VehicleInterfaces.Examples.SeriesHybrid

Series hybrid vehicle model



Information

Example model architecture of a passenger car with a series hybrid powertrain. In this architecture variant the PowerTrain systems are rigidly mounted to the chassis when MultiBody effects are included.

Extends from [VehicleInterfaces.Examples.PartialVehicle](#) (Basic model for a four wheeled vehicle, requires powertrain components).

Parameters

Name	Description
Plant Models	
chassis	Chassis subsystem
driveline	Driveline subsystem
brakes	Brakes subsystem
engine	Engine model
accessories	Accessories model
generator	Generator subsystem
driveMotor	Traction motor subsystem
battery	Energy storage subsystem
Conditions	
road	Road model
atmosphere	Atmospheric model
world	Global co-ordinate system

VehicleInterfaces.Examples.PartialVehicle

Basic model for a four wheeled vehicle, requires powertrain components

**Information**

Base model architecture of a passenger car that can be extended to add any type of powertrain. In this architecture variant the Driveline subsystem is rigidly mounted to the chassis when MultiBody effects are included.

Extends from Modelica.Icons.Example (Icon for runnable examples).

Parameters

Name	Description
Plant Models	
chassis	Chassis subsystem
driveline	Driveline subsystem
brakes	Brakes subsystem
Driver Models	
driverEnvironment	Driver environment
Conditions	
road	Road model
atmosphere	Atmospheric model
world	Global co-ordinate system

VehicleInterfaces.Blocks

Collection of input/output fixed causality blocks

Information

A collection of additional input and output fixed causality blocks.

Extends from Modelica.Icons.Package (Icon for standard packages).

Package Content

Name	Description
 RealPassThrough	Pass a Real signal through without modification
 IntegerPassThrough	Pass a Integer signal through without modification
 BooleanPassThrough	Pass a Boolean signal through without modification
 InvertNormalizedInput	Output the normalized input signal u $[0..1]$ in inverted form y $[1..0]$.

VehicleInterfaces.Blocks.RealPassThrough

Pass a Real signal through without modification



Information

Passes a Real signal through without modification. Enables signals to be read out of one bus, have their name changed and be sent back to a bus.

Extends from Modelica.Blocks.Interfaces.BlockIcon (Basic graphical layout of input/output block).

Connectors

Name	Description
u	Input signal
y	Output signal

VehicleInterfaces.Blocks.IntegerPassThrough

Pass a Integer signal through without modification



Information

Passes a Integer signal through without modification. Enables signals to be read out of one bus, have their name changed and be sent back to a bus.

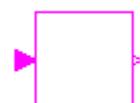
Extends from Modelica.Blocks.Interfaces.IntegerBlockIcon (Basic graphical layout of Integer block).

Connectors

Name	Description
u	Input signal
y	Output signal

VehicleInterfaces.Blocks.BooleanPassThrough

Pass a Boolean signal through without modification



Information

Passes a Boolean signal through without modification. Enables signals to be read out of one bus, have their name changed and be sent back to a bus.

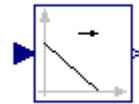
Extends from Modelica.Blocks.Interfaces.BooleanBlockIcon (Basic graphical layout of Boolean block).

Connectors

Name	Description
u	Input signal
y	Output signal

VehicleInterfaces.Blocks.InvertNormalizedInput

Output the normalized input signal u [0..1] in inverted form y [1..0].



Information

This block is used to invert a normalized input signal u from [0..1] to [1..0]. Formally, the output y is computed as:

$$y = 1 - u;$$

Extends from Modelica.Blocks.Interfaces.SISO (Single Input Single Output continuous control block).

Connectors

Name	Description
u	Connector of Real input signal
y	Connector of Real output signal

VehicleInterfaces.Icons

Collection of icons

Information

A collection of basic icon definitions used in the package

Extends from Modelica.Icons.Package (Icon for standard packages).

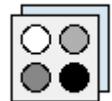
Package Content

Name	Description
 VariantLibrary	Icon for a package class which contains several variants of one assembly or component
 BaseClassPackage	Icon for a package class named BaseClasses or Internal
 Accessories	Icon for an accessories subsystem
 Atmosphere	Icon for an atmosphere
 Battery	Icon for a battery subsystem
 Brakes	Icon for a brakes subsystem

 Chassis	Icon for a chassis subsystem
 Controller	Icon for a controller subsystem
 DataDictionary	Icon for a data dictionary
 Driveline	Icon for a driveline subsystem
 DriverEnvironment	Icon for a driver environment subsystem
 Driver	Icon for a driver subsystem
 ElectricMotor	Icon for an electric drive subsystem
 Empty	Icon for an empty component
 Engine	Icon for an engine subsystem
 MultipleMounts	Icon for a multiple mounting subsystem
 Road	Icon for a road
 SingleMount	Icon for a single mounting subsystem
 TwoMounts	Icon for a multiple mounting subsystem
 Transmission	Icon for a transmission subsystem
 SignalSubBusWithExplicitSignals	Icon for signal sub-bus where the explicit signals are defined in the bus

VehicleInterfaces.Icons.VariantLibrary

Icon for a package class which contains several variants of one assembly or component



Information

This partial class is intended to design a *default icon for a package class* which contains several *variants* of one assembly or component.

Extends from Modelica.Icons.VariantsPackage (Icon for package containing variants).

VehicleInterfaces.Icons.BaseClassPackage

Icon for a package class named **BaseClasses** or **Internal**



Information

This partial class is intended to design a *default icon for a package class* which is called *BaseClasses* or *Internal*.

VehicleInterfaces.Icons.Accessories

Icon for an accessories subsystem



Information

This partial class is intended to design a default icon for an *accessories assembly*.

VehicleInterfaces.Icons.Atmosphere

Icon for an atmosphere

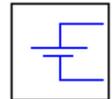


Information

This partial class is intended to design a default icon for an *atmosphere*.

VehicleInterfaces.Icons.Battery

Icon for a battery subsystem

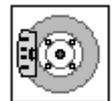


Information

This partial class is intended to design a default icon for a *battery model*.

VehicleInterfaces.Icons.Brakes

Icon for a brakes subsystem



Information

This partial class is intended to design a default icon for a *brakes assembly*.

VehicleInterfaces.Icons.Chassis

Icon for a chassis subsystem

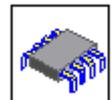


Information

This partial class is intended to design a default icon for a *chassis assembly*.

VehicleInterfaces.Icons.Controller

Icon for a controller subsystem

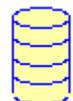


Information

This partial class is intended to design a default icon for *controllers*.

VehicleInterfaces.Icons.DataDictionary

Icon for a data dictionary

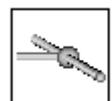


Information

This partial class is intended to design a default icon for a *data dictionary*.

VehicleInterfaces.Icons.Driveline

Icon for a driveline subsystem



Information

This partial class is intended to design a default icon for a *driveline assembly*.

VehicleInterfaces.Icons.DriverEnvironment

Icon for a driver environment subsystem



Information

This partial class is intended to design a default icon for a *driver environment*.

VehicleInterfaces.Icons.Driver

Icon for a driver subsystem

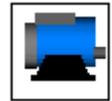


Information

This partial class is intended to design a default icon for a *driver model*.

VehicleInterfaces.Icons.ElectricMotor

Icon for an electric drive subsystem



Information

This partial class is intended to design a default icon for an *electric motor*.

VehicleInterfaces.Icons.Empty

Icon for an empty component



Information

This partial class is intended to design a default icon for an *empty element*, i.e. for element which has no influence on the overall model.

VehicleInterfaces.Icons.Engine

Icon for an engine subsystem

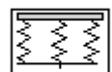


Information

This partial class is intended to design a default icon for a *engine assembly*.

VehicleInterfaces.Icons.MultipleMounts

Icon for a multiple mounting subsystem



Information

This partial class is intended to design a default icon for a model of *multiple mounts*.

VehicleInterfaces.Icons.Road

Icon for a road

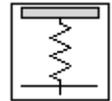


Information

This partial class is intended to design a default icon for a *road*.

VehicleInterfaces.Icons.SingleMount

Icon for a single mounting subsystem

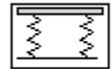


Information

This partial class is intended to design a default icon for a model of *single mount*.

VehicleInterfaces.Icons.TwoMounts

Icon for a multiple mounting subsystem

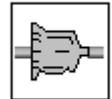


Information

This partial class is intended to design a default icon for a model of *two mounts*.

VehicleInterfaces.Icons.Transmission

Icon for a transmission subsystem



Information

This partial class is intended to design a default icon for a *transmission assembly*.

VehicleInterfaces.Icons.SignalSubBusWithExplicitSignals

Icon for signal sub-bus where the explicit signals are defined in the bus

Information

This icon is designed for a **sub-bus** in a signal connector.

VehicleInterfaces.Interfaces

Collection of general interface definitions

Information

This library provides general interface definitions, such as the signal bus.

Extends from Modelica.Icons.InterfacesPackage (Icon for packages containing interfaces).

Package Content

Name	Description
------	-------------

 ControlBus	Bus of VehicleInterfaces.Interfaces: Minimal standard control bus
 AccessoriesBus	Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as accessory bus
 AccessoriesControlBus	Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as accessories control bus
 BatteryBus	Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as battery bus
 BatteryControlBus	Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as battery control bus
 BrakesBus	Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as brakes bus
 BrakesControlBus	Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as brakes control bus
 ChassisBus	Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as chassis bus
 ChassisControlBus	Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as chassis control bus
 DrivelineBus	Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as driveline bus
 DrivelineControlBus	Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as driveline control bus
 DriverBus	Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as driver bus
 DriverInterface	Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as driver interface bus
 ElectricMotorBus	Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as an electric motor bus
 ElectricMotorControlBus	Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as an electric motor control bus
 EngineBus	Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as engine bus
 EngineControlBus	Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as engine control bus
 TransmissionBus	Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as transmission bus
 TransmissionControlBus	Bus of VehicleInterfaces.Interfaces: Bus that contains a minimal set of signals generated by the Transmission Controller model
 ShiftConnector	Manual transmission shift connector
 ShiftInput	Convert an integer signal to a position in the manual shift connector
 ShiftOutput	Convert a position in the manual shift connector to an integer value

VehicleInterfaces.Interfaces.ControlBus

Bus of VehicleInterfaces.Interfaces: Minimal standard control bus



Information

An empty expandable connector used as the top-level control signal bus in VehicleInterfaces.

Extends from Modelica.Icons.SignalBus (Icon for signal bus).

Parameters

Name	Description
accessoriesBus	Accessories bus
accessoriesControlBus	Accessories control bus
batteryBus	Battery bus
brakesBus	Brakes bus
brakesControlBus	Brakes control bus
chassisBus	Chassis bus
chassisControlBus	Chassis control bus
drivelineBus	Driveline bus
drivelineControlBus	Driveline control bus
driverBus	Driver bus
electricMotorBus	Electric motor bus
electricMotorControlBus	Electric motor control bus
engineBus	Engine bus
engineControlBus	Engine control bus
transmissionBus	Transmission bus
transmissionControlBus	Transmission control bus

Contents

Name	Description
accessoriesBus	Accessories bus
accessoriesControlBus	Accessories control bus
batteryBus	Battery bus
brakesBus	Brakes bus
brakesControlBus	Brakes control bus
chassisBus	Chassis bus
chassisControlBus	Chassis control bus
drivelineBus	Driveline bus
drivelineControlBus	Driveline control bus
driverBus	Driver bus
electricMotorBus	Electric motor bus
electricMotorControlBus	Electric motor control bus
engineBus	Engine bus
engineControlBus	Engine control bus
transmissionBus	Transmission bus
transmissionControlBus	Transmission control bus

VehicleInterfaces.Interfaces.AccessoriesBus

Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as accessory bus



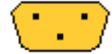
Information

Should be used to contain signals measured in the accessories subsystem. It is defined as an empty expandable connector.

Extends from Modelica.Icons.SignalSubBus (Icon for signal sub-bus).

VehicleInterfaces.Interfaces.AccessoriesControlBus

Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as accessories control bus



Information

Should be used to contain signals determined in the accessories controller. It is defined as an empty expandable connector.

Extends from Modelica.Icons.SignalSubBus (Icon for signal sub-bus).

VehicleInterfaces.Interfaces.BatteryBus

Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as battery bus



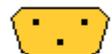
Information

Should be used to contain signals measured in the battery subsystem. It is defined as an empty expandable connector.

Extends from Modelica.Icons.SignalSubBus (Icon for signal sub-bus).

VehicleInterfaces.Interfaces.BatteryControlBus

Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as battery control bus



Information

Should be used to contain signals measured in the battery subsystem. It is defined as an empty expandable connector.

Extends from Modelica.Icons.SignalSubBus (Icon for signal sub-bus).

VehicleInterfaces.Interfaces.BrakesBus

Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as brakes bus



Information

Should be used to contain signals measured in the brakes subsystem. It is defined as an empty expandable connector.

Extends from Modelica.Icons.SignalSubBus (Icon for signal sub-bus).

VehicleInterfaces.Interfaces.BrakesControlBus

Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as brakes control bus



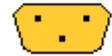
Information

Should be used to contain signals determined in the brakes controller. It is defined as an empty expandable connector.

Extends from Modelica.Icons.SignalSubBus (Icon for signal sub-bus).

VehicleInterfaces.Interfaces.ChassisBus

Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as chassis bus



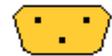
Information

Should be used to contain signals measured in the chassis subsystem. It is defined as an empty expandable connector.

Extends from Modelica.Icons.SignalSubBus (Icon for signal sub-bus).

VehicleInterfaces.Interfaces.ChassisControlBus

Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as chassis control bus



Information

Should be used to contain signals determined in the chassis controller. It is defined as an empty expandable connector.

Extends from Modelica.Icons.SignalSubBus (Icon for signal sub-bus).

VehicleInterfaces.Interfaces.DrivelineBus

Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as driveline bus



Information

Should be used to contain signals measured in the driveline subsystem. It is defined as an empty expandable connector.

Extends from Modelica.Icons.SignalSubBus (Icon for signal sub-bus).

VehicleInterfaces.Interfaces.DrivelineControlBus

Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as driveline control bus



Information

Should be used to contain signals determined in the driveline controller. It is defined as an empty expandable connector.

Extends from Modelica.Icons.SignalSubBus (Icon for signal sub-bus).

VehicleInterfaces.Interfaces.DriverBus

Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as driver bus



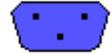
Information

Should be used to contain signals that are determined in the driverEnvironment subsystem. It is defined as an empty expandable connector.

Extends from Modelica.Icons.SignalSubBus (Icon for signal sub-bus).

VehicleInterfaces.Interfaces.DriverInterface

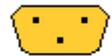
Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as driver interface bus

**Information**

Should be used to contain signals and normalised mechanical connections that need to be exchanged between the Driver and DriverEnvironment subsystems.

VehicleInterfaces.Interfaces.ElectricMotorBus

Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as an electric motor bus

**Information**

Should be used to contain signals measured in the electrical motor subsystem. It is defined as an empty expandable connector.

Extends from Modelica.Icons.SignalSubBus (Icon for signal sub-bus).

VehicleInterfaces.Interfaces.ElectricMotorControlBus

Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as an electric motor control bus

**Information**

Should be used to contain signals measured in the electrical motor control subsystem. It is defined as an empty expandable connector.

Extends from Modelica.Icons.SignalSubBus (Icon for signal sub-bus).

VehicleInterfaces.Interfaces.EngineBus

Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as engine bus

**Information**

Should be used to contain signals measured in the engine subsystem. It is defined as an empty expandable connector.

Extends from Modelica.Icons.SignalSubBus (Icon for signal sub-bus).

VehicleInterfaces.Interfaces.EngineControlBus

Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as engine control bus



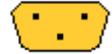
Information

Should be used to contain signals determined in the engine controller. It is defined as an empty expandable connector.

Extends from Modelica.Icons.SignalSubBus (Icon for signal sub-bus).

VehicleInterfaces.Interfaces.TransmissionBus

Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as transmission bus



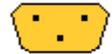
Information

Should be used to contain signals measured in the transmission subsystem. It is defined as an empty expandable connector.

Extends from Modelica.Icons.SignalSubBus (Icon for signal sub-bus).

VehicleInterfaces.Interfaces.TransmissionControlBus

Bus of VehicleInterfaces.Interfaces: Bus that contains a minimal set of signals generated by the Transmission Controller model



Information

Should be used to contain signals determined in the transmission controller. It is defined as an empty expandable connector.

Extends from Modelica.Icons.SignalSubBus (Icon for signal sub-bus).

Parameters

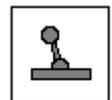
Name	Description
currentGear	Currently selected gear

Contents

Name	Description
currentGear	Currently selected gear

VehicleInterfaces.Interfaces.ShiftConnector

Manual transmission shift connector



Information

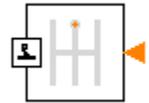
Manual transmission shift connector. It contains two 1D translational connectors that represent the position of the shift lever across the shift gate (**crossGate**) and forwards/backwards within the gate (**intoGear**).

Contents

Name	Description
crossGate	Gear shift lever movement across the selection gate
intoGear	Gear shift lever movement forwards and backwards in the selection gate

VehicleInterfaces.Interfaces.ShiftInput

Convert an integer signal to a position in the manual shift connector



Information

Converts an integer signal in to positions on the **intoGear** and **crossGate** sub-connectors of the ShiftConnector.

Parameters

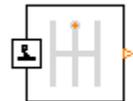
Name	Description
numberOfGears	Number of forward gears
crossGateDisplacement	Total distance across the gate [m]
intoGearDisplacement	Distance to move in to gear [m]

Connectors

Name	Description
shiftConnector	Gear shift connection
gear	Gear number

VehicleInterfaces.Interfaces.ShiftOutput

Convert a position in the manual shift connector to an integer value



Information

Converts positions on the **intoGear** and **crossGate** sub-connectors of the ShiftConnector into an integer gear signal.

Parameters

Name	Description
numberOfGears	Number of forward gears
crossGateDisplacement	Total distance across the gate [m]
intoGearDisplacement	Distance to move in to gear [m]

Connectors

Name	Description
shiftConnector	Gear shift connection
gear	Gear number

VehicleInterfaces.Mechanics

Collection of additional mechanical components

Information

Additional mechanical components required by the library and models based on this package. Includes 1D mechanical connectors and components with normalized position and angle variables rather than actual

values. Also includes the MultiBodyEnd component which applies zero force and torque at the end of a chain of components.

Extends from Modelica.Icons.Package (Icon for standard packages).

Package Content

Name	Description
 NormalisedTranslational	Collection of normalized translational mechanics
 NormalisedRotational	Collection of normalized rotational mechanics
 MultiBody	Collection of additional MultiBody components

VehicleInterfaces.Mechanics.NormalisedTranslational

Collection of normalized translational mechanics

Information

Additional 1D translational mechanical components with normalized position.

Extends from Modelica.Icons.Package (Icon for standard packages).

Package Content

Name	Description
 Interfaces	Collection of connector definitions
 Position	Applies a position to a Flange
 Force	Applies a force to a Flange
 PositionSensor	Ideal sensor to measure the position of a Flange

VehicleInterfaces.Mechanics.NormalisedTranslational.Interfaces

Collection of connector definitions

Information

A collection of additional 1D translational mechanical connectors used for normalized translational elements.

Extends from Modelica.Icons.InterfacesPackage (Icon for packages containing interfaces).

Package Content

Name	Description
 Flange	1D normalized translational flange

VehicleInterfaces.Mechanics.NormalisedTranslational.Interfaces.Flange

1D normalized translational flange



Information

1D translational mechanical connector with normalized position as opposed to the [Modelica.Mechanics.Translational.Interfaces.flange_a](#) connector. The force is not normalized in this connector.

Contents

Name	Description
s	Normalized position of flange [1]
f	Cut force directed into flange [N]

VehicleInterfaces.Mechanics.NormalisedTranslational.Position

Applies a position to a Flange



Information

Applies a normalized position to a 1D normalized translational system. Cannot be connected to components from the Modelica.Mechanics.Translational library

Parameters

Name	Description
exact	True/false exact treatment/filtering the input signal
f_crit	If exact=false: critical frequency of filter to filter input signal [Hz]

Connectors

Name	Description
flange_b	flange to be positioned
position	position to be applied

VehicleInterfaces.Mechanics.NormalisedTranslational.Force

Applies a force to a Flange



Information

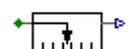
Applies a force to a 1D normalized translational system. Cannot be connected to components from the Modelica.Mechanics.Translational library

Connectors

Name	Description
flange_b	Flange to be forced
force	Force to be applied

VehicleInterfaces.Mechanics.NormalisedTranslational.PositionSensor

Ideal sensor to measure the position of a Flange



Information

Measures a normalized position in a 1D normalized translational system. Cannot be connected to components from the Modelica.Mechanics.Translational library

Extends from Modelica.Icons.TranslationalSensor (Icon representing a linear measurement device).

Connectors

Name	Description
flange_a	Flange to be measured
position	Absolute position of flange as output signal

VehicleInterfaces.Mechanics.NormalisedRotational**Collection of normalized rotational mechanics****Information**

Additional 1D rotational mechanical components with normalized angle of rotation.

Extends from Modelica.Icons.Package (Icon for standard packages).

Package Content

Name	Description
 Interfaces	Collection of connector definitions
 Position	Applies a rotation to a Flange
 Torque	Applies a torque to a Flange
 AngleSensor	Ideal sensor to measure the rotation of a Flange

VehicleInterfaces.Mechanics.NormalisedRotational.Interfaces**Collection of connector definitions****Information**

A collection of additional 1D rotational mechanical connectors used for normalized translational elements.

Extends from Modelica.Icons.InterfacesPackage (Icon for packages containing interfaces).

Package Content

Name	Description
 Flange	1D normalized rotational flange

VehicleInterfaces.Mechanics.NormalisedRotational.Interfaces.Flange**1D normalized rotational flange****Information**

1D rotational mechanical connector with normalized position as opposed to the

Modelica.Mechanics.Rotational.Interfaces.Flange_a connector. The torque is not normalized in this connector.

Contents

Name	Description
phi	Normalized rotation of flange [1]
tau	Cut torque directed in the flange [N.m]

VehicleInterfaces.Mechanics.NormalisedRotational.Position

Applies a rotation to a Flange



Information

Applies a normalized angle to a 1D normalized rotational system. Cannot be connected to components from the Modelica.Mechanics.Rotational library

Parameters

Name	Description
exact	True/false exact treatment/filtering the input signal
f_crit	If exact=false: critical frequency of filter to filter input signal [Hz]

Connectors

Name	Description
flange_b	flange to be rotated
phi_ref	position to be applied

VehicleInterfaces.Mechanics.NormalisedRotational.Torque

Applies a torque to a Flange



Information

Applies a torque to a 1D normalized rotational system. Cannot be connected to components from the Modelica.Mechanics.Rotational library

Connectors

Name	Description
flange_b	Flange to be turned
tau	Torque to be applied

VehicleInterfaces.Mechanics.NormalisedRotational.AngleSensor

Ideal sensor to measure the rotation of a Flange



Information

Measures a normalized angle in a 1D normalized rotational system. Cannot be connected to components

from the Modelica.Mechanics.Rotational library

Extends from Modelica.Icons.RotationalSensor (Icon representing a round measurement device).

Connectors

Name	Description
flange_a	flange to be measured
phi	Absolute angle of flange as output signal

VehicleInterfaces.Mechanics.MultiBody

Collection of additional MultiBody components

Information

Multi-body components required by the library which are not included in Modelica Standard Library (package Modelica.Mechanics.MultiBody).

Extends from Modelica.Icons.Package (Icon for standard packages).

Package Content

Name	Description
 MultiBodyEnd	Ends a MultiBody chain

VehicleInterfaces.Mechanics.MultiBody.MultiBodyEnd

Ends a MultiBody chain



Information

This component is used to end a chain of multi-body components by applying zero force and torque to the end of the chain. Uses the FlangeWithBearing connector where the **flange** connector has zero torque applied to it and the **bearingFrame** can be optionally included through the parameter `includeBearingConnector`. If `includeBearingConnector=true` then zero force and torque is applied to the bearingFrame.

Parameters

Name	Description
includeBearingConnector	= true, if bearing frame connector is present, otherwise not present

Connectors

Name	Description
flange	Flange with zero force and torque applied

VehicleInterfaces.Types

Collection of type definitions

Information

Type definitions required by the library.

Extends from Modelica.Icons.Package (Icon for standard packages).

Package Content

Name	Description
 NormalizedReal	Normalized real number
 Gear	Gear ..., -2, -1, 0, 1, 2, ... = 2nd rear, 1st rear, neutral, 1st, 2nd forward, etc
 GearMode	Type, constants and menu choices for gear mode, as temporary solution until enumerations are available
 IgnitionSetting	Type, constants and menu choices for ignition setting, as temporary solution until enumerations are available

VehicleInterfaces.Types.NormalizedReal

Normalized real number



Parameters

Name	Description
quantity	
unit	
displayUnit	
min	
max	

VehicleInterfaces.Types.Gear

Gear ..., -2, -1, 0, 1, 2, ... = 2nd rear, 1st rear, neutral, 1st, 2nd forward, etc



Parameters

Name	Description
quantity	

VehicleInterfaces.Types.GearMode

Type, constants and menu choices for gear mode, as temporary solution until enumerations are available

Information

Extends from Modelica.Icons.Package (Icon for standard packages).

Package Content

Name	Description
Park=1	

Drive=2	
Neutral=3	
Rear=4	
Manual=5	
Limited=6	
Temp	Temporary type of gear mode with choices for menus (until enumerations are available)

VehicleInterfaces.Types.GearMode.Temp

Temporary type of gear mode with choices for menus (until enumerations are available)

Information

Extends from Integer.

Parameters

Name	Description
min	
max	

VehicleInterfaces.Types.IgnitionSetting

Type, constants and menu choices for ignition setting, as temporary solution until enumerations are available

Information

Extends from Modelica.Icons.Package (Icon for standard packages).

Package Content

Name	Description
Off=1	Ignition is off
On=2	Ignition is on
Start=3	Ignition is on and driver requires to start engine
Temp	Temporary type of ignition setting with choices for menus (until enumerations are available)

VehicleInterfaces.Types.IgnitionSetting.Temp

Temporary type of ignition setting with choices for menus (until enumerations are available)

Information

Extends from Integer.

Parameters

Name	Description
min	

max	
-----	--

VehicleInterfaces.Accessories

Collection of accessory subsystem definitions

Information

The accessory subsystem interfaces are defined in this sub-package of the VehicleInterfaces library. The accessories subsystem has the following connectors:

- **engineFlange** - 1D rotational connection to the engine subsystem (or other systems mechanically connected to the accessories)
- **controlBus** - control signal bus connection

Effects to be modelled in this subsystem

Within the VehicleInterfaces package the accessories subsystem is used to model the torque losses and inertia effects of the engine accessories such as the alternator, power steering pump, etc.

Extends from [VehicleInterfaces.Icons.VariantLibrary](#) (Icon for a package class which contains several variants of one assembly or component).

Package Content

Name	Description
 Tutorial	Accessories Tutorial
 Interfaces	Collection of interface definitions for accessories
 NoAccessories	Empty accessories
 MinimalAccessories	Constant torque loss and inertia due to the accessories

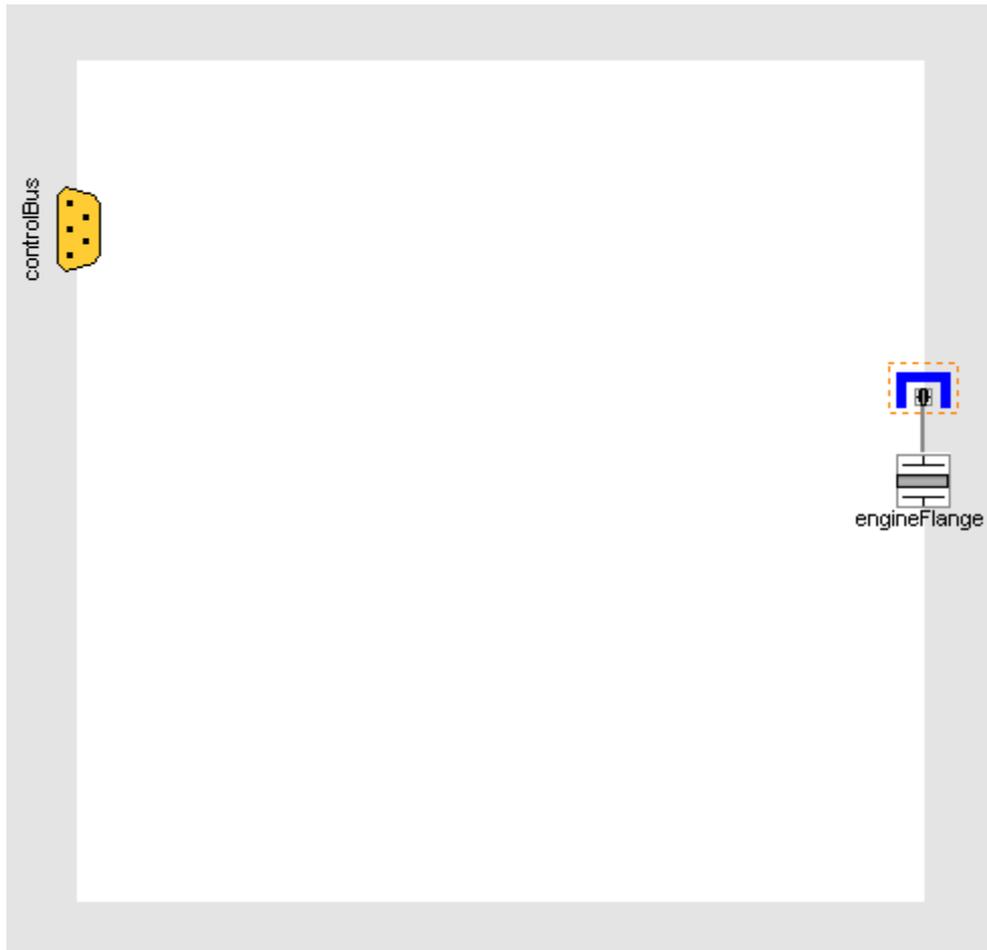
VehicleInterfaces.Accessories.Tutorial

Tutorial - Defining a new accessories model

The following process will demonstrate how to create a new accessories model using these interface definitions. This tutorial will guide you through building an accessory subsystem with a power steering pump and alternator. Both will be modelled using speed dependant torque loss maps.



1. Create a new model that extends **VehicleInterfaces.Accessories.Interfaces.Base**, it should look like this:



2. In the component browser, right click on **Base** and select **Parameters** from the context menu to produce the following parameter dialog:



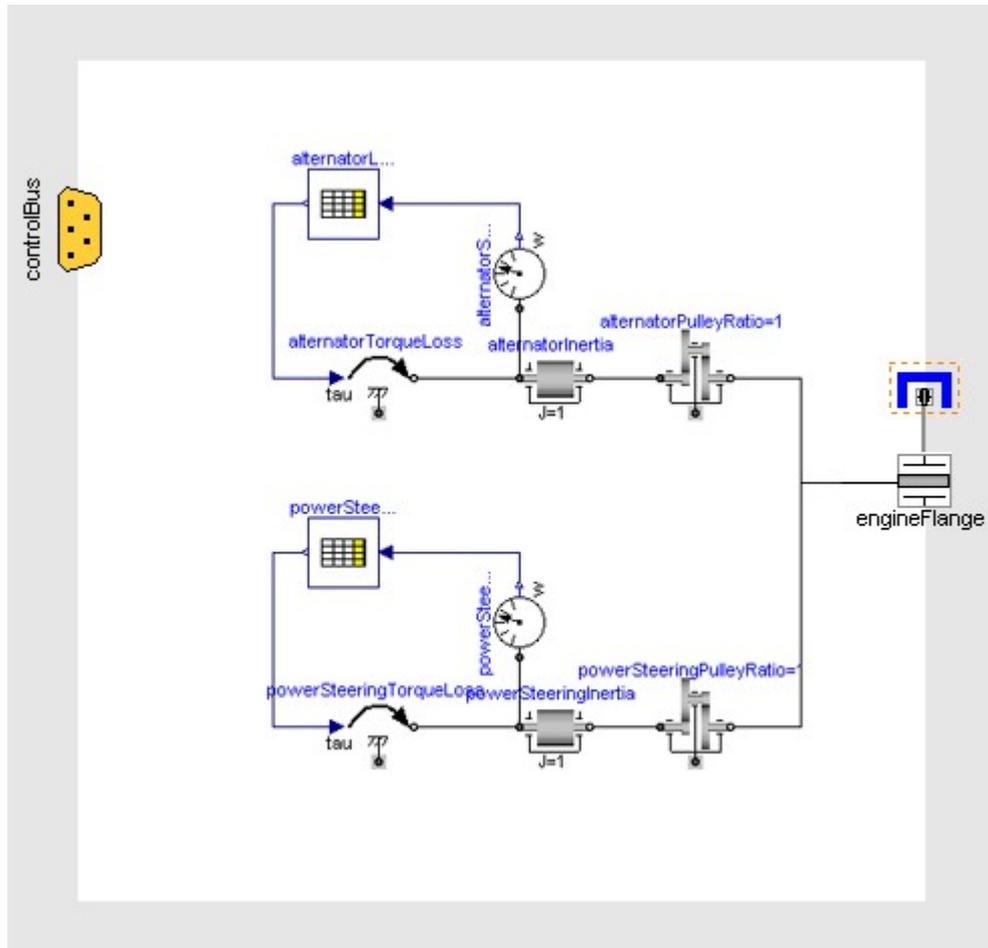
3. This dialog allows you to enable/disable the optional bearing connectors within the engineFlange by setting **includeEngineBearing** as required.
4. You can now define your accessory model as required

Adding an alternator and power steering pump

The following steps demonstrate how to create a simple accessories model. The subsystem will include an alternator and power steering pump that are modelled separately but both use speed-dependant torque loss maps.

Starting from step 2 above.

1. Add the following blocks and connections to the diagram.



2. Next, we need to check to see if any connections to the control signal bus are required for the driveline, see [a complete list](#) of the minimum connections required. In this case we don't need to add any signals to the control signal bus.
3. The model is now complete and should check successfully and can be used in any model compatible with the VehicleInterfaces library

VehicleInterfaces.Accessories.Interfaces

Collection of interface definitions for accessories

Information

A collection of partial base classes which define interfaces for accessories models.

Extends from Modelica.Icons.InterfacesPackage (Icon for packages containing interfaces).

Package Content

Name	Description
▪ Base	Basic interface definition for the accessories

VehicleInterfaces.Accessories.Interfaces.Base

Basic interface definition for the accessories

Information

This partial model defines the interfaces required for an accessories subsystem within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Parameters

Name	Description
Advanced	
usingMultiBodyEngine	=true if using a MultiBody engine with a 1D accessories model

Connectors

Name	Description
engineFlange	Source of drive torque
controlBus	Control signal bus

VehicleInterfaces.Accessories.NoAccessories

Empty accessories



Information

Empty accessories subsystem. Using this empty model in overall vehicle architecture the functionality of accessories subsystem can be eliminated.

Extends from [VehicleInterfaces.Icons.Accessories](#) (Icon for an accessories subsystem), [VehicleInterfaces.Icons.Empty](#) (Icon for an empty component), [Interfaces.Base](#) (Basic interface definition for the accessories).

Parameters

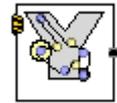
Name	Description
Advanced	
usingMultiBodyEngine	=true if using a MultiBody engine with a 1D accessories model

Connectors

Name	Description
engineFlange	Source of drive torque
controlBus	Control signal bus

VehicleInterfaces.Accessories.MinimalAccessories

Constant torque loss and inertia due to the accessories



Information

Simple accessory model with a single inertia and constant torque loss.

Extends from [VehicleInterfaces.Icons.Accessories](#) (Icon for an accessories subsystem), [Interfaces.Base](#) (Basic interface definition for the accessories).

Parameters

Name	Description
accessoriesInertia	Total effective rotational inertia of the accessories [kg.m2]
accessoriesLoad	Total effective torque load due to the accessories (constant) [N.m]
axisOfRotation	Axis of rotation for accessories when including MultiBody effects [1]
Advanced	
usingMultiBodyEngine	=true if using a MultiBody engine with a 1D accessories model

Connectors

Name	Description
engineFlange	Source of drive torque
controlBus	Control signal bus

VehicleInterfaces.Atmospheres

Collection of atmosphere subsystem definitions

Information

The atmosphere subsystem interfaces are defined in this sub-package of the VehicleInterfaces library. The atmosphere subsystem has no connectors but contains standard functions that can be redeclared to implement different ambient conditions. When an atmosphere subsystem is used in a model architecture it should be declared as an **inner** system so that it's functions can be accessed by other components in the model.

Effects to be modelled in this subsystem

Within the VehicleInterfaces package the atmosphere subsystem is used to define the ambient conditions. These include air temperature, pressure, density, humidity and wind speed.

Extends from [VehicleInterfaces.Icons.VariantLibrary](#) (Icon for a package class which contains several variants of one assembly or component).

Package Content

Name	Description
 Tutorial	Atmosphere Tutorial
 Interfaces	Collection of interface definitions for atmosphere
 ConstantAtmosphere	Atmospheres with constant conditions

VehicleInterfaces.Atmospheres.Tutorial

Tutorial - Defining a new atmosphere model



This tutorial will guide you through the definition of the **ConstantAtmosphere** model.

1. Start creating a new model by extending the base atmosphere definition which can be found at **VehicleInterfaces.Atmospheres.Interfaces.Base**
2. This base atmosphere definition includes 4 partial function definitions that we need to provide complete functions for in our new atmosphere model. The following functions need to be defined:
 - windVelocity
 - density
 - temperature
 - humidity
3. Determine the model assumptions to be used. We will create a model with constant atmospheric conditions, i.e. fixed pressure, temperature, humidity and a constant wind. Create the following functions within the model created in step 1.
4. Define the windVelocity function. This function is used to define the wind velocity based on the position relative to the world axis:

```
function constantWindVelocity
  extends VehicleInterfaces.Atmospheres.Interfaces.Base.windVelocity;
  input Modelica.SIunits.Velocity[3] windVelocity;
algorithm
  v := windVelocity;
end constantWindVelocity;
```

5. Define the density function. This function is used to define the density based on the position relative to the world axis:

```
function constantDensity "Determine density"
  extends VehicleInterfaces.Atmospheres.Interfaces.Base.density;
  input Modelica.SIunits.Density density "Density";
algorithm
  rho := density;
end constantDensity;
```

6. Define the temperature function. This function is used to define the temperature based on the position relative to the world axis:

```
function constantTemperature "Determine temperature"
  extends VehicleInterfaces.Atmospheres.Interfaces.Base.temperature;
  input Modelica.SIunits.Temperature T0 "Constant temperature";
algorithm
  T := T0;
end constantTemperature;
```

7. Define the humidity function. This function is used to define the humidity based on the position relative to the world axis:

```
function constantHumidity "Determine humidity"
  extends VehicleInterfaces.Atmospheres.Interfaces.Base.humidity;
  input Real h0 "Constant humidity";
algorithm
  h := h0;
```

```
end constantTemperature;
```

8. We now need to redeclare the functions in the base atmosphere definition to be the functions shown above. We do this with the following code assuming that the above functions have been defined within the class **ConstantAtmosphere**:

```
model ConstantAtmosphere "Atmosphere with constant conditions"
  extends VehicleInterfaces.Atmospheres.Interfaces.Base(
    redeclare final function windVelocity = constantWindVelocity
(windVelocity=v),
    redeclare final function density = constantDensity (density=rho),
    redeclare final function temperature = constantTemperature (T0=T),
    redeclare final function humidity = constantHumidity(h0=h));

  //Rest of model definition
  ...
end ConstantAtmosphere;
```

9. Adding these redeclares also means we need to add 4 parameters to the model **v**, **rho**, **h** and **T** which are the constant wind velocity, density, humidity and temperature respectively.
10. This model is now complete and can be used. The ConstantAtmosphere definition included in VehicleInterfaces actually has pressure, temperature and wind velocity as it's parameters and automatically calculates the air density.

VehicleInterfaces.Atmospheres.Interfaces

Collection of interface definitions for atmosphere

Information

A collection of base classes which define interfaces for atmosphere models.

Extends from Modelica.Icons.Package (Icon for standard packages).

Package Content

Name	Description
 windVelocityBase	Determine wind velocity
 densityBase	Determine density
 temperatureBase	Determine temperature
 humidityBase	Determine humidity
Base	Base model for all atmospheres

VehicleInterfaces.Atmospheres.Interfaces.windVelocityBase

Determine wind velocity



Information

Partial base model for wind velocity. Extend this model appropriately to define final user model.

Extends from Modelica.Icons.Function (Icon for functions).

Inputs

Name	Description
r[3]	Position vector from world frame to point, resolved in world frame [m]

Outputs

Name	Description
v[3]	Wind velocity vector, resolved in world frame [m/s]

VehicleInterfaces.Atmospheres.Interfaces.densityBase**Determine density****Information**

Partial base model for air density. Extend this model appropriately to define final user model.

Extends from Modelica.Icons.Function (Icon for functions).

Inputs

Name	Description
r[3]	Position vector from world frame to point, resolved in world frame [m]

Outputs

Name	Description
rho	Density [kg/m ³]

VehicleInterfaces.Atmospheres.Interfaces.temperatureBase**Determine temperature****Information**

Partial base model for air temperature. Extend this model appropriately to define final user model.

Extends from Modelica.Icons.Function (Icon for functions).

Inputs

Name	Description
r[3]	Position vector from world frame to point, resolved in world frame [m]

Outputs

Name	Description
T	Temperature [K]

VehicleInterfaces.Atmospheres.Interfaces.humidityBase**Determine humidity**

Information

Partial base model for air humidity. Extend this model appropriately to define final user model.

Extends from Modelica.Icons.Function (Icon for functions).

Inputs

Name	Description
r[3]	Position vector from world frame to point, resolved in world frame [m]

Outputs

Name	Description
h	Humidity

VehicleInterfaces.Atmospheres.Interfaces.Base

Base model for all atmospheres

Information

This partial model defines the interfaces required for an atmosphere subsystem within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

VehicleInterfaces.Atmospheres.ConstantAtmosphere

Atmospheres with constant conditions



Information

This is the simplest atmosphere model with constant characteristics.

Extends from [VehicleInterfaces.Icons.Atmosphere](#) (Icon for an atmosphere),
[VehicleInterfaces.Atmospheres.Interfaces.Base](#) (Base model for all atmospheres).

Parameters

Name	Description
replaceable function windVelocity	Dummy model for wind velocity
replaceable function density	Dummy model for air density
replaceable function temperature	Dummy model for air temperature
replaceable function humidity	Dummy model for air humidity
v[3]	Wind velocity [m/s]
ambientPressure	Air pressure [Pa]
T	Air temperature [K]
h	Air humidity (0-1)

Connectors

Name	Description
replaceable function windVelocity	Dummy model for wind velocity
replaceable function density	Dummy model for air density

replaceable function temperature	Dummy model for air temperature
replaceable function humidity	Dummy model for air humidity

VehicleInterfaces.Brakes

Collection of brakes subsystem definitions

Information

The brakes subsystem interfaces are defined in this sub-package of the VehicleInterfaces library. The brakes subsystem has the following connectors some of which are optional (see below for more information):

- **controlBus** - control signal bus connection
- **wheelHub_n** - wheelHub connectors that consist of a 1D rotational connector and a MultiBody frame connector (see [here](#). The number of these varies depending on how many wheels the vehicle has.
- **brakePedal** - 1D translational connection for the brake pedal connection to the driverEnvironment (optional)

The optional connectors are, by default, disabled and can be ignored if not required. They can be enabled by setting the appropriate parameter to be true. This is only possible at design time, i.e. when you are building the subsystem model.

Effects to be modelled in this subsystem

Within the VehicleInterfaces package the brakes subsystem is used to model the entire braking system from the brake pedal position through to the torque applied at the wheel hubs to decelerate the vehicle and the reactions in to the wheel carrier. Different interface definitions are provided for vehicles with different numbers of wheels, a FlangeWithBearing connector is added for each wheel.

Extends from [VehicleInterfaces.Icons.VariantLibrary](#) (Icon for a package class which contains several variants of one assembly or component).

Package Content

Name	Description
 Tutorial	Brakes Tutorial
 Interfaces	Collection of interface definitions for brakes
 NoBrakes	Empty brakes model for 4 wheeled vehicles
 MinimalBrakes	Simple proportional (constant gain) braking model for 4 wheeled vehicles
 MinimalBrakesUsingPedal	Simple proportional (constant gain) braking model for 4 wheeled vehicles, uses the brake pedal connection

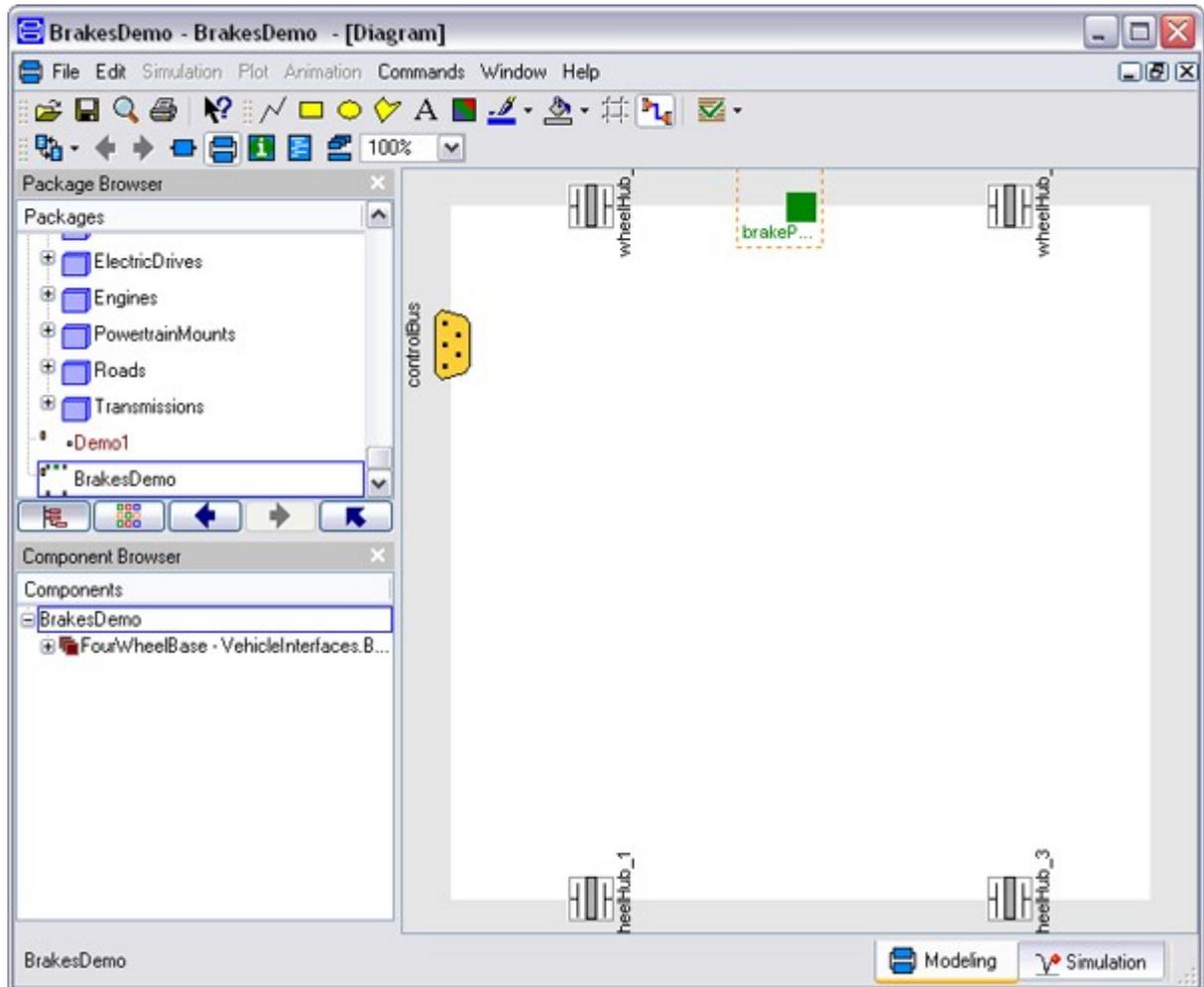
VehicleInterfaces.Brakes.Tutorial

Tutorial - Defining a new brakes model

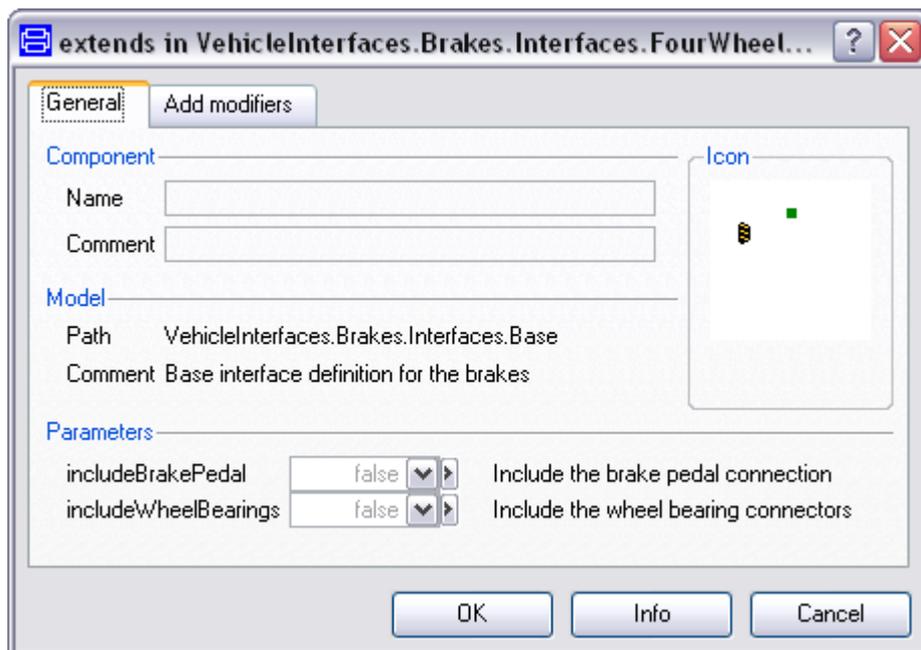
The following process will demonstrate how to create a new brakes model using these interface definitions. This tutorial will guide you through building a braking system for a passenger car, i.e. a vehicle with 4 wheels.

1. Create a new model that extends **VehicleInterfaces.Brakes.Interfaces.FourWheelBase**, it should look like this:





- In the component browser, right click on **FourWheelBase** and select **Parameters** from the context menu to produce the following parameter dialog



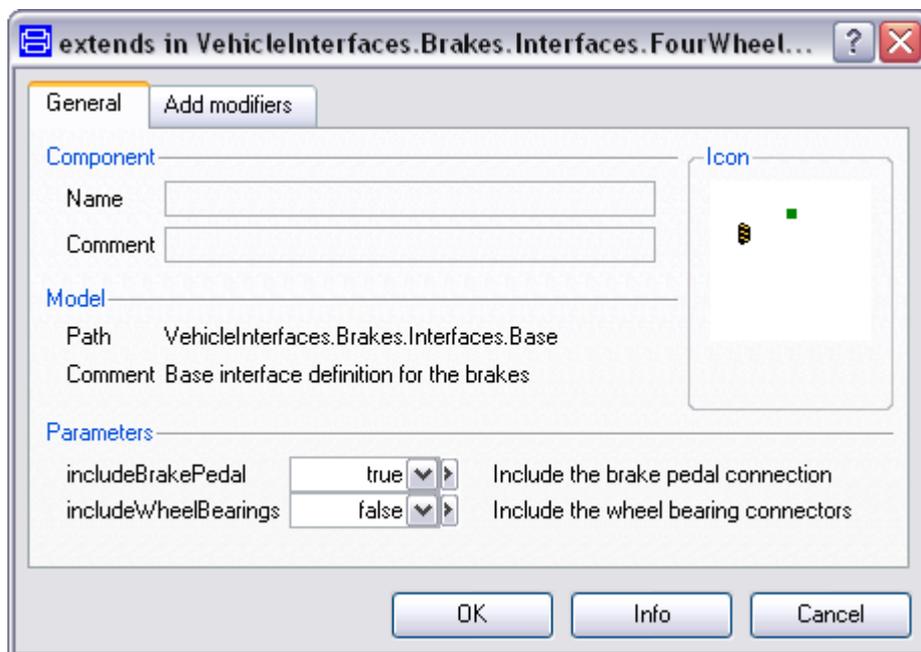
3. This dialog allows you to enable/disable the optional connections by setting **includeWheelBearings** and **includeBrakePedal** as required for your new brakes model. The wheelHub connectors are of the type [VehicleInterfaces.Interfaces.FlangeWithBearing](#), the parameter **includeWheelBearings** controls whether the bearing connectors within the wheelHubs is enabled or not.
4. You can now define your brakes model as required

Creating a simple braking system example

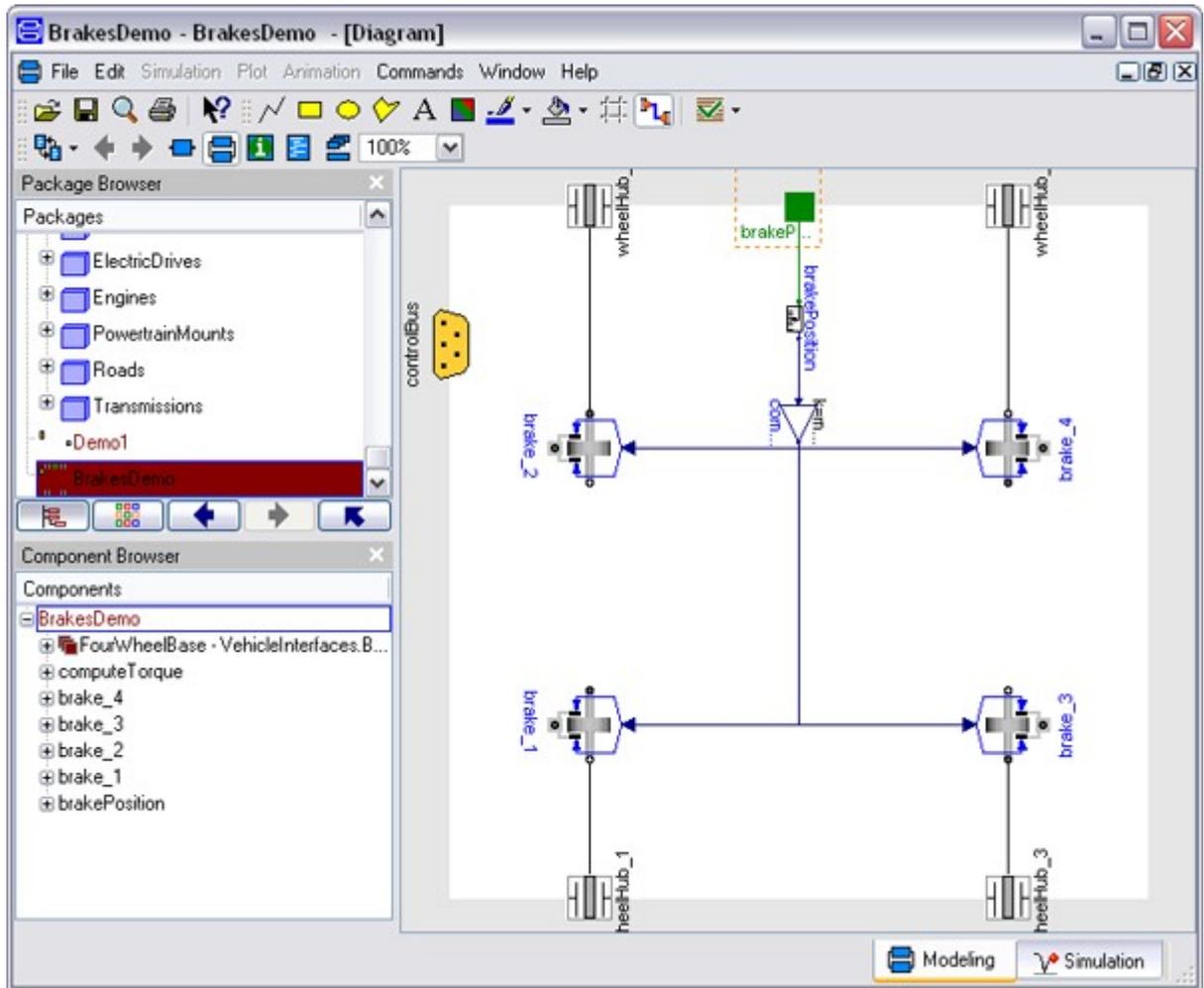
The following steps demonstrate how to create a simple braking system model. The brakes will be modelled using the standard brakes component from the Modelica standard library. A position sensor will measure the brake pedal position and a 1D table block will be used to convert this to a force that will be applied to each wheel brake. The reactions in to the wheel carriers will not be modelled.

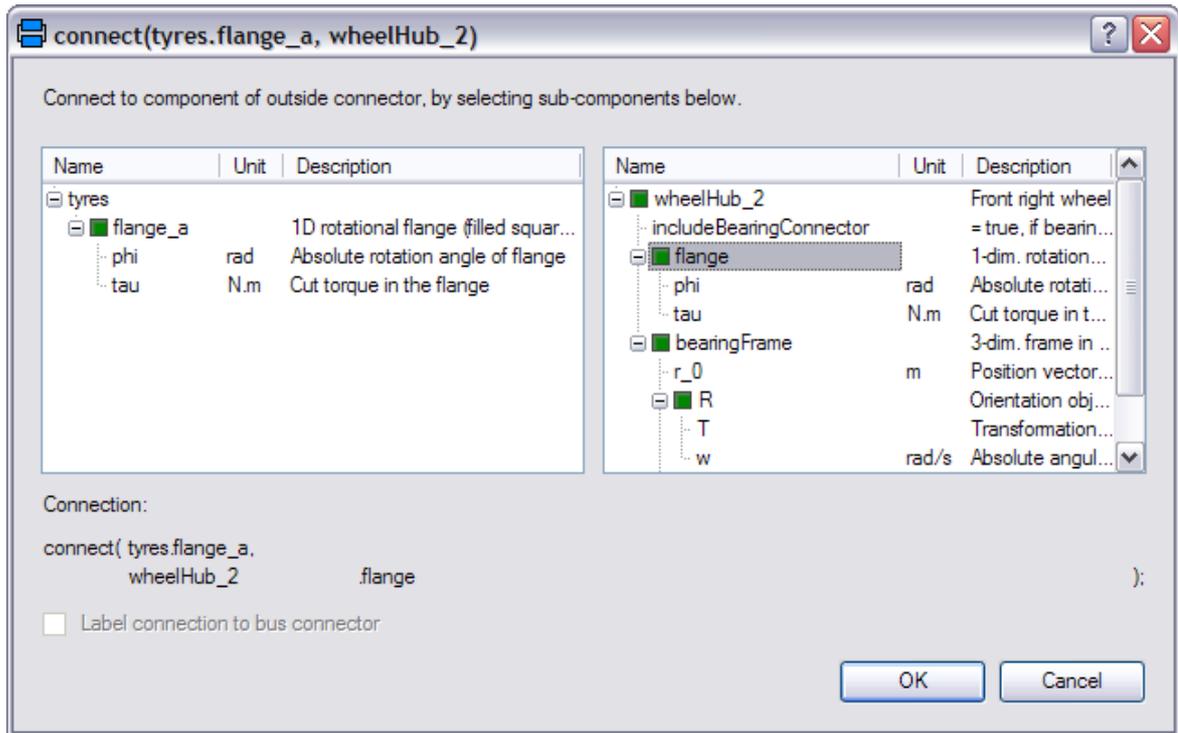
Starting from step 3 above.

1. First, decide which of the optional connectors are required in the model. For this example we need the brake pedal connection but not the wheel bearing connector

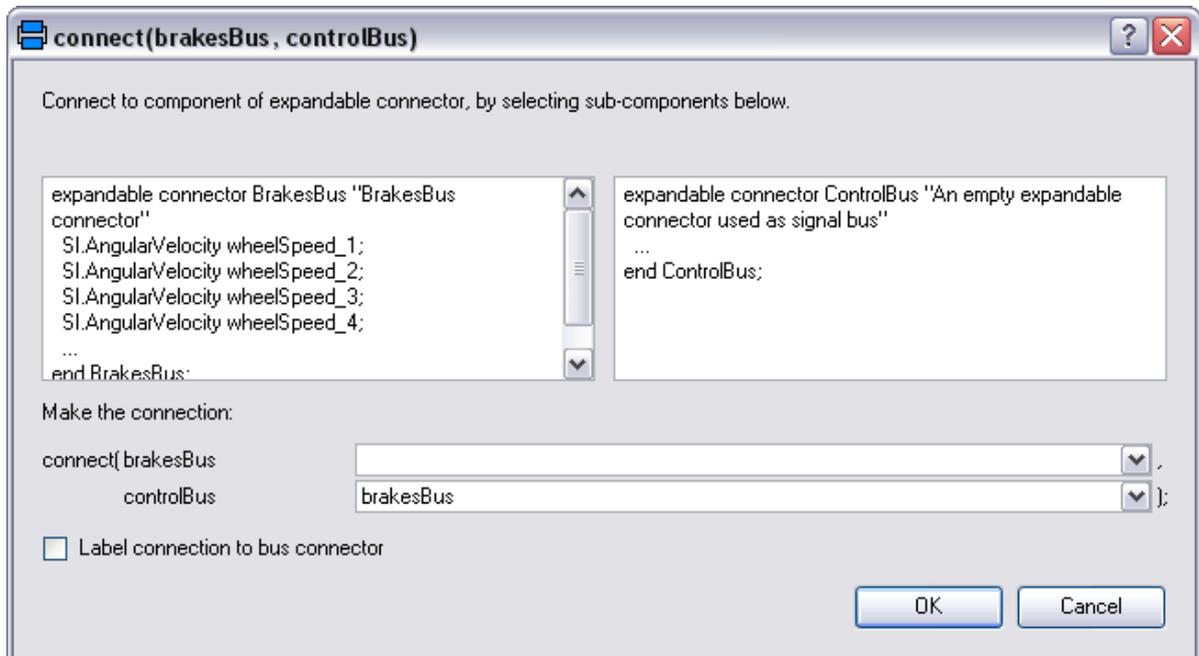


2. Add the following blocks and connections to the diagram. When you draw the connections from the brake components to the wheelHub connectors the dialog box shown below will appear asking which connector within the wheelHub connector you would like to make the connection to. As we are modelling the brakes as a 1D system you should select **flange** from the list of options which is the 1D connector within the wheelHub connector.

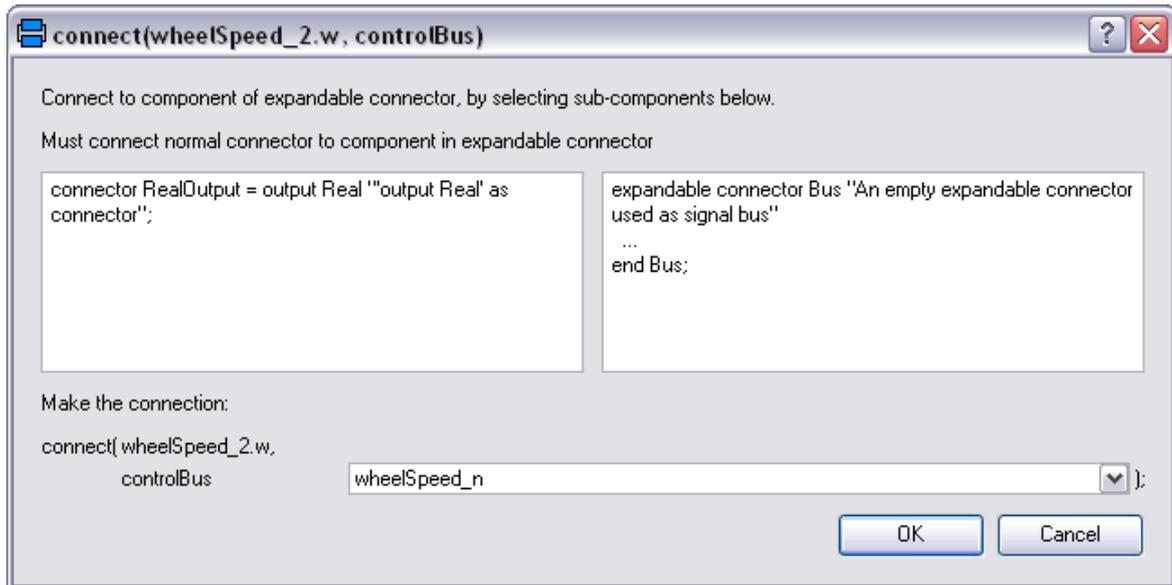




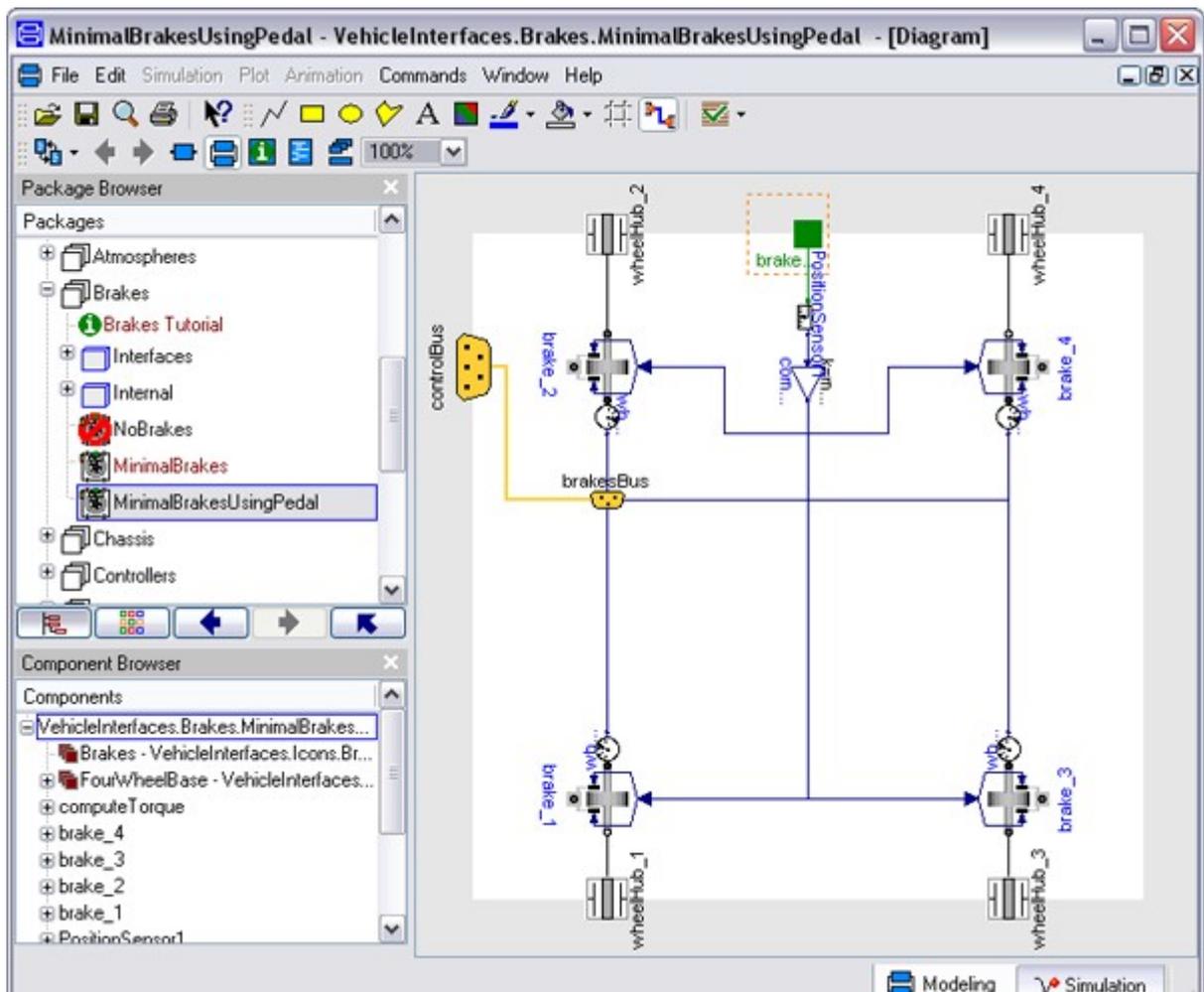
3. Next, we need to check to see if any connections to the control signal bus are required for the brakes, see [here](#) for a complete list of the minimum connections required. In this case we need to add the four individual wheel speeds to the control signal bus and this can be done by connecting speed sensors to each wheel and then connecting these to the signal bus. As the wheel speed signals are added to the brakesBus we first need to add this connector. The brakesBus connector is **VehicleInterfaces.Interfaces.BrakesBus**. This should be connected to the **controlBus**, when this connection is made the following dialog is produced and should be completed as shown.



4. When the connections between the sensors and the brakesBus connector are added the dialog below appears and should be completed replacing **n** with the correct index for the wheel speed being measured



- The model is now complete and should check successfully and can be used in any model compatible with the VehicleInterfaces library assuming the selected Driver model also uses the brake pedal connection



VehicleInterfaces.Brakes.Interfaces

Collection of interface definitions for brakes

Information

A collection of partial base classes which define interfaces for brakes models.

Extends from Modelica.Icons.InterfacesPackage (Icon for packages containing interfaces).

Package Content

Name	Description
 Base	Base interface definition for a brakes
 TwoAxleBase	Interface definition for the brakes on a 4 wheeled vehicle
 ThreeAxleBase	Interface definition for the brakes on a 6 wheeled vehicle
 FourAxleBase	Interface definition for the brakes on a 8 wheeled vehicle
 StandardBus	Bus of VehicleInterfaces.Brakes: StandardBus of signals generated by the brakes

VehicleInterfaces.Brakes.Interfaces.Base

Base interface definition for a brakes

Information

This partial model defines the basic interfaces required for any brakes subsystem within the VehicleInterfaces package. This class should be extended to form a brakes interface definition with the correct number of wheelHub connectors for the type of vehicle being modelled. See the [documentation](#) and [tutorial](#) for more information.

Parameters

Name	Description
Advanced	
usingMultiBodyChassis	=true if using a MultiBody chassis with a 1D driveline

Connectors

Name	Description
controlBus	Control signal bus
brakePedal	Brake pedal connection (optional)

VehicleInterfaces.Brakes.Interfaces.TwoAxleBase

Interface definition for the brakes on a 4 wheeled vehicle

Information

This partial model defines the interfaces required for the brakes subsystem of a two axled vehicle within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Extends from [Base](#) (Base interface definition for a brakes).

Parameters

Name	Description
Advanced	
usingMultiBodyChassis	=true if using a MultiBody chassis with a 1D driveline

Connectors

Name	Description
controlBus	Control signal bus
brakePedal	Brake pedal connection (optional)
wheelHub_1	Front left wheel
wheelHub_2	Front right wheel
wheelHub_3	Rear left wheel
wheelHub_4	Rear right wheel

VehicleInterfaces.Brakes.Interfaces.ThreeAxleBase

Interface definition for the brakes on a 6 wheeled vehicle



Information

This partial model defines the interfaces required for the brakes subsystem of a three axled vehicle within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Extends from [Base](#) (Base interface definition for a brakes).

Parameters

Name	Description
Advanced	
usingMultiBodyChassis	=true if using a MultiBody chassis with a 1D driveline

Connectors

Name	Description
controlBus	Control signal bus
brakePedal	Brake pedal connection (optional)
wheelHub_1	Front left wheel
wheelHub_2	Front right wheel
wheelHub_3	Rear left wheel
wheelHub_4	Rear right wheel
wheelHub_5	Rear left wheel
wheelHub_6	Rear right wheel

VehicleInterfaces.Brakes.Interfaces.FourAxleBase

Interface definition for the brakes on a 8 wheeled vehicle



Information

This partial model defines the interfaces required for the brakes subsystem of a four axled vehicle within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Extends from [Base](#) (Base interface definition for a brakes).

Parameters

Name	Description
Advanced	
usingMultiBodyChassis	=true if using a MultiBody chassis with a 1D driveline

Connectors

Name	Description
controlBus	Control signal bus
brakePedal	Brake pedal connection (optional)
wheelHub_1	Front left wheel
wheelHub_2	Front right wheel
wheelHub_3	Rear left wheel
wheelHub_4	Rear right wheel
wheelHub_5	Rear left wheel
wheelHub_6	Rear right wheel
wheelHub_7	Rear left wheel
wheelHub_8	Rear right wheel

VehicleInterfaces.Brakes.Interfaces.StandardBus

Bus of VehicleInterfaces.Brakes: StandardBus of signals generated by the brakes



Information

Bus with a set of standard signals generated by the brakes subsystem.

Extends from [.VehicleInterfaces.Interfaces.BrakesBus](#) (Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as brakes bus), [.VehicleInterfaces.Icons.SignalSubBusWithExplicitSignals](#) (Icon for signal sub-bus where the explicit signals are defined in the bus).

Parameters

Name	Description
wheelSpeed_1	Individual wheel speed (wheel 1 = front left wheel) [rad/s]
wheelSpeed_2	Individual wheel speed (wheel 2 = front right wheel) [rad/s]
wheelSpeed_3	Individual wheel speed (wheel 3 = rear left wheel) [rad/s]
wheelSpeed_4	Individual wheel speed (wheel 4 = rear right wheel) [rad/s]

Contents

Name	Description
wheelSpeed_1	Individual wheel speed (wheel 1 = front left wheel) [rad/s]

wheelSpeed_2	Individual wheel speed (wheel 2 = front right wheel) [rad/s]
wheelSpeed_3	Individual wheel speed (wheel 3 = rear left wheel) [rad/s]
wheelSpeed_4	Individual wheel speed (wheel 4 = rear right wheel) [rad/s]

VehicleInterfaces.Brakes.NoBrakes

Empty brakes model for 4 wheeled vehicles



Information

Zero torque is applied to all the wheelhubs and the reaction paths in to the wheel hubs included if the **driveTrainMechanics3D** flag in the world object is true.

Using this empty model in overall vehicle architecture the functionality of brakes can be eliminated.

Extends from [VehicleInterfaces.Icons.Brakes](#) (Icon for a brakes subsystem), [VehicleInterfaces.Icons.Empty](#) (Icon for an empty component), [VehicleInterfaces.Brakes.Interfaces.TwoAxleBase](#) (Interface definition for the brakes on a 4 wheeled vehicle).

Parameters

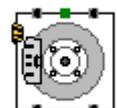
Name	Description
Advanced	
usingMultiBodyChassis	=true if using a MultiBody chassis with a 1D driveline

Connectors

Name	Description
controlBus	Control signal bus
brakePedal	Brake pedal connection (optional)
wheelHub_1	Front left wheel
wheelHub_2	Front right wheel
wheelHub_3	Rear left wheel
wheelHub_4	Rear right wheel

VehicleInterfaces.Brakes.MinimalBrakes

Simple proportional (constant gain) braking model for 4 wheeled vehicles



Information

Brakes subsystem model that uses the driver **brakePedalPosition** signal to determine the brake torque demand being requested by the driver.

Extends from [VehicleInterfaces.Icons.Brakes](#) (Icon for a brakes subsystem), [VehicleInterfaces.Brakes.Interfaces.TwoAxleBase](#) (Interface definition for the brakes on a 4 wheeled vehicle).

Parameters

Name	Description
maxTorque	Maximum combined brake torque (for all brakes together) [N.m]
Advanced	

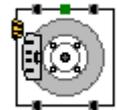
usingMultiBodyChassis	=true if using a MultiBody chassis with a 1D driveline
-----------------------	--

Connectors

Name	Description
controlBus	Control signal bus
brakePedal	Brake pedal connection (optional)
wheelHub_1	Front left wheel
wheelHub_2	Front right wheel
wheelHub_3	Rear left wheel
wheelHub_4	Rear right wheel

VehicleInterfaces.Brakes.MinimalBrakesUsingPedal

Simple proportional (constant gain) braking model for 4 wheeled vehicles, uses the brake pedal connection



Information

Brakes subsystem model that uses the physical pedal connection to determine the brake torque demand being requested by the driver.

Extends from [VehicleInterfaces.Icons.Brakes](#) (Icon for a brakes subsystem), [VehicleInterfaces.Brakes.Interfaces.TwoAxleBase](#) (Interface definition for the brakes on a 4 wheeled vehicle).

Parameters

Name	Description
maxTorque	Maximum combined brake torque [N.m]
Advanced	
usingMultiBodyChassis	=true if using a MultiBody chassis with a 1D driveline

Connectors

Name	Description
controlBus	Control signal bus
brakePedal	Brake pedal connection (optional)
wheelHub_1	Front left wheel
wheelHub_2	Front right wheel
wheelHub_3	Rear left wheel
wheelHub_4	Rear right wheel

VehicleInterfaces.Chassis

Collection of chassis subsystem definitions

Information

The chassis subsystem interfaces are defined in this sub-package of the VehicleInterfaces library. The chassis subsystem has the following connectors some of which are optional (see below for more

information):

- **controlBus** - control signal bus connection
- **wheelHub_n** - wheelHub connectors that consist of a 1D rotational connector and a MultiBody frame connector (see [here](#)). The number of these varies depending on how many wheels the vehicle has.
- **steeringWheel** - 1D rotational connection for the steering wheel connection to the driverEnvironment (optional)
- **chassisMount** - MultiBody connection providing a connection point to the vehicle body (optional)

The optional connectors are, by default, disabled and can be ignored if not required. They can be enabled by setting the appropriate parameter to be true. This is only possible at design time, i.e. when you are building the subsystem model.

Effects to be modelled in this subsystem

Within the VehicleInterfaces package the chassis subsystem is used to model the wheels, tires, suspension and vehicle body.

Extends from [VehicleInterfaces.Icons.VariantLibrary](#) (Icon for a package class which contains several variants of one assembly or component).

Package Content

Name	Description
 Tutorial	Chassis Tutorial
 Interfaces	Collection of interface definitions for chassis
 NoChassis	Empty chassis
 MinimalChassis	Basic chassis model with rigid connection between all 4 wheels
 MinimalChassis2	Basic chassis model with rigid connection between all 4 wheels, uses chassisFrame connection
 MinimalChassis3	Basic chassis model with rigid connection between all 4 wheels, uses chassisFrame connection, includes wheel bearings

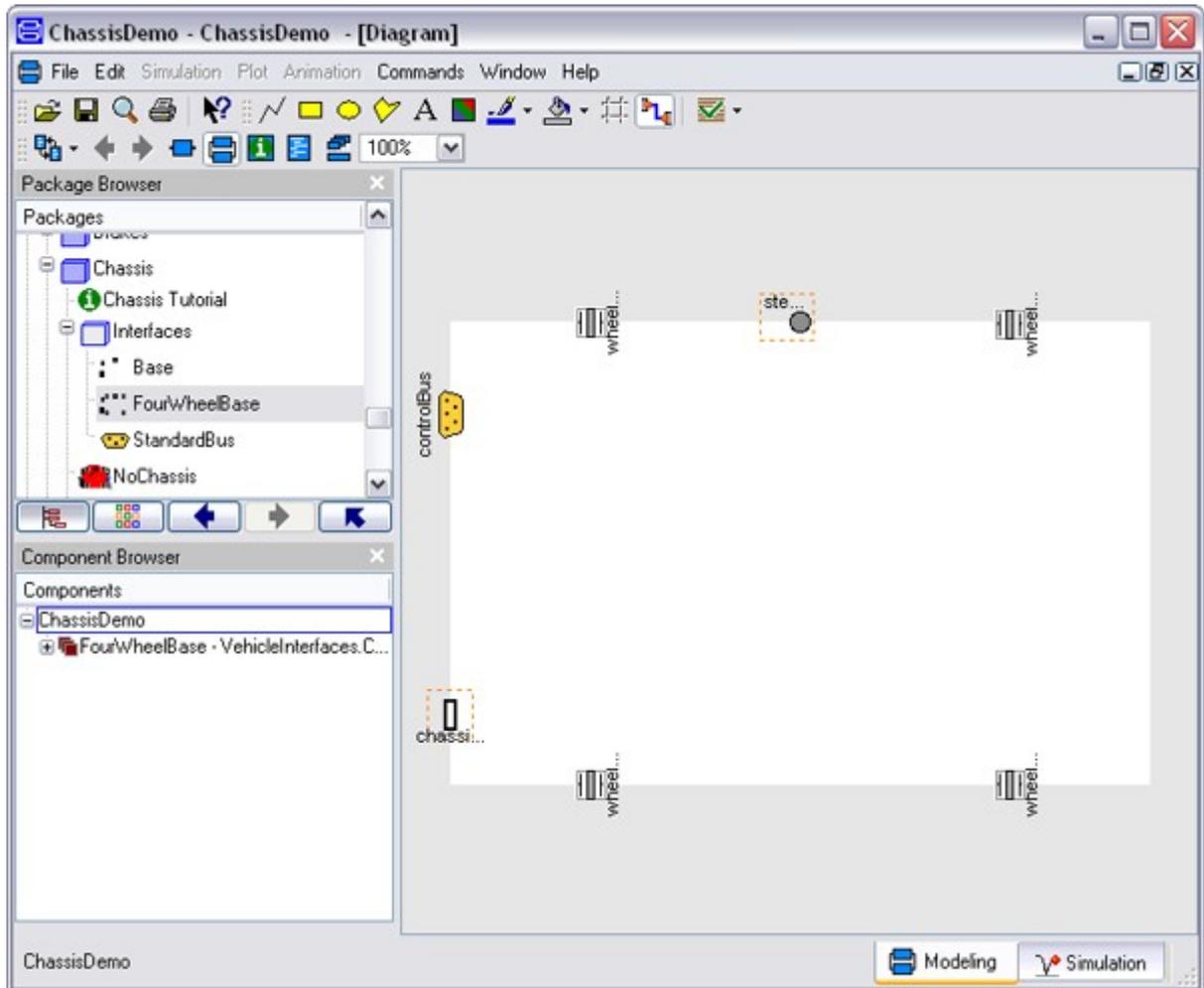
VehicleInterfaces.Chassis.Tutorial

Tutorial - Defining a new chassis model

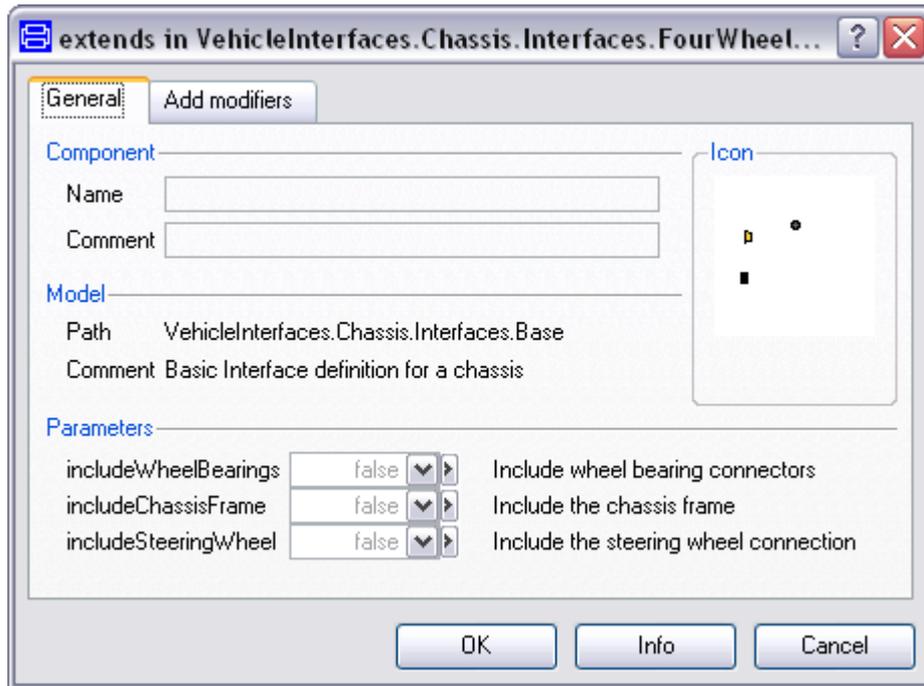
The following process will demonstrate how to create a new chassis model using these interface definitions. This tutorial will guide you through building a chassis for a passenger car, i.e. a vehicle with 4 wheels.



1. Create a new model that extends **VehicleInterfaces.Chassis.Interfaces.FourWheelBase**, it should look like this:



2. In the component browser, right click on **FourWheelBase** and select **Parameters** from the context menu to produce the following parameter dialog



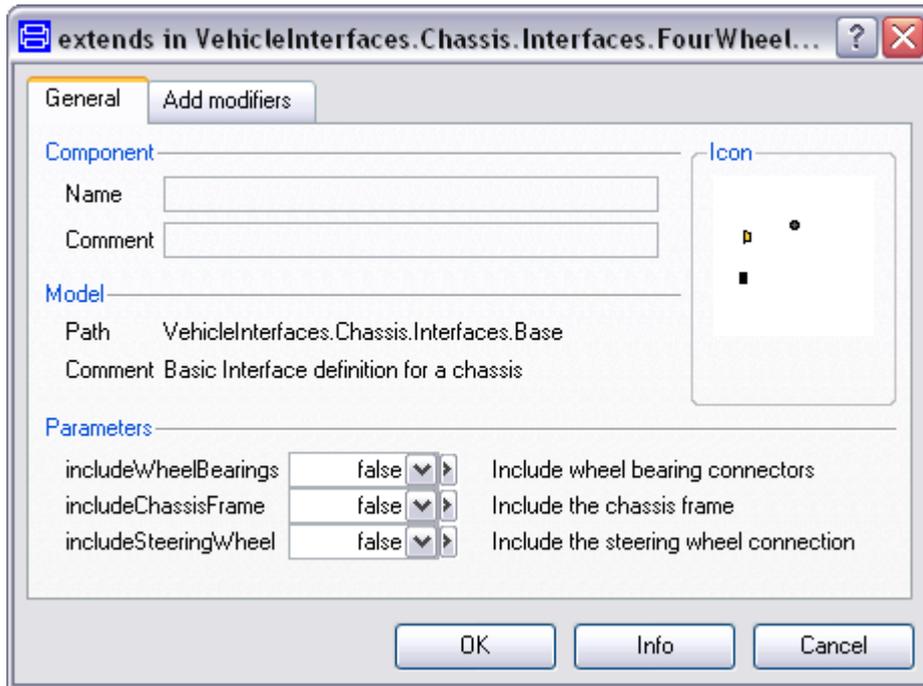
3. This dialog allows you to enable/disable the optional connections by setting **includeWheelBearings**, **includeSteeringWheel** and **includeChassisFrame** as required for your new chassis model. The wheelHub connectors are of the type [VehicleInterfaces.Interfaces.FlangeWithBearing](#), the parameter **includeWheelBearings** controls whether the bearing connectors within the wheelHubs is enabled or not.
4. You can now define your chassis model as required

Creating a simple chassis model for longitudinal motion

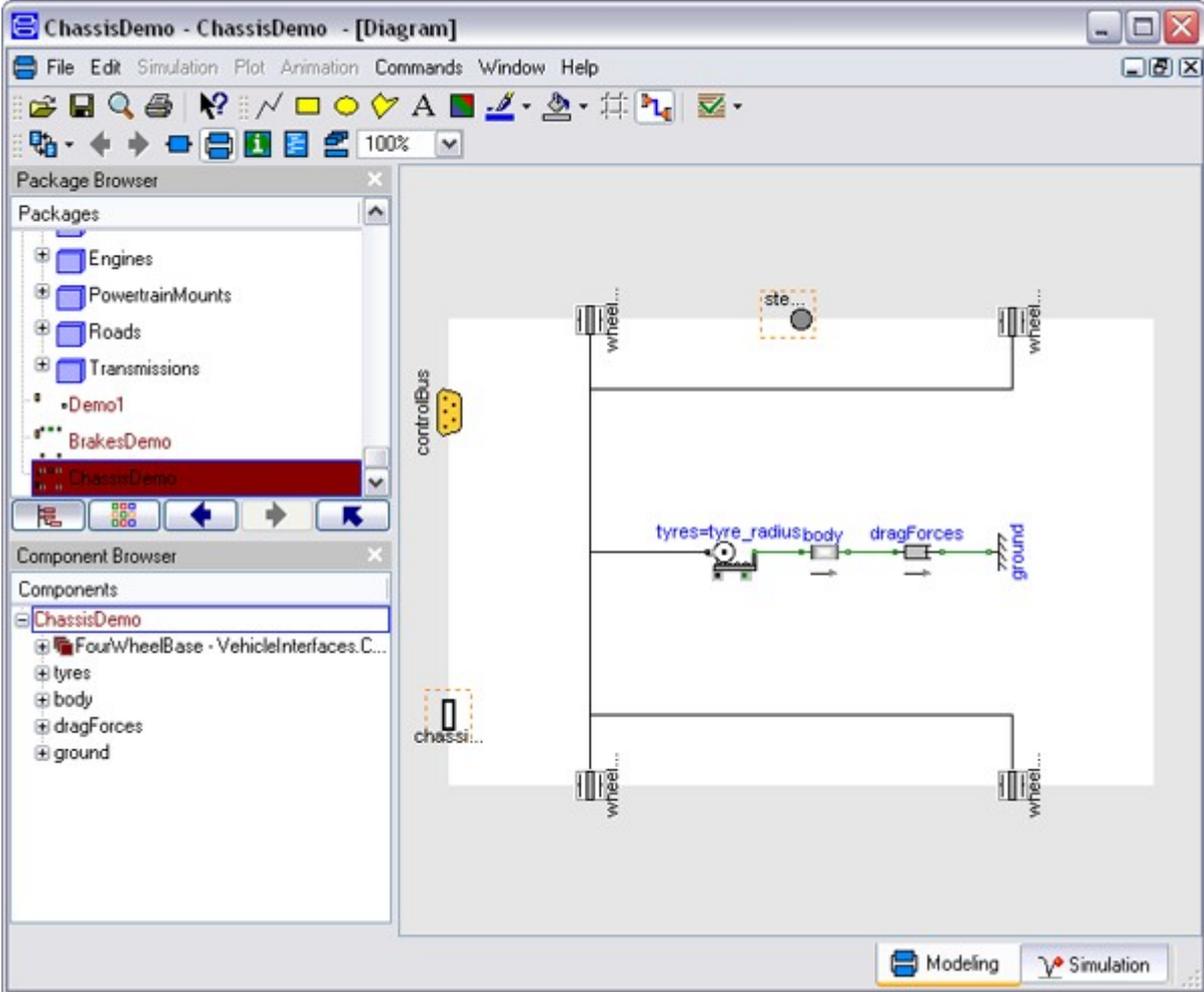
The following steps demonstrate how to create a simple chassis model for longitudinal motion. The chassis model will transmit the torque from the wheel hub connections in to a force being applied to the vehicle model. Only the longitudinal dynamics of the vehicle body will be modelled so no suspension is required.

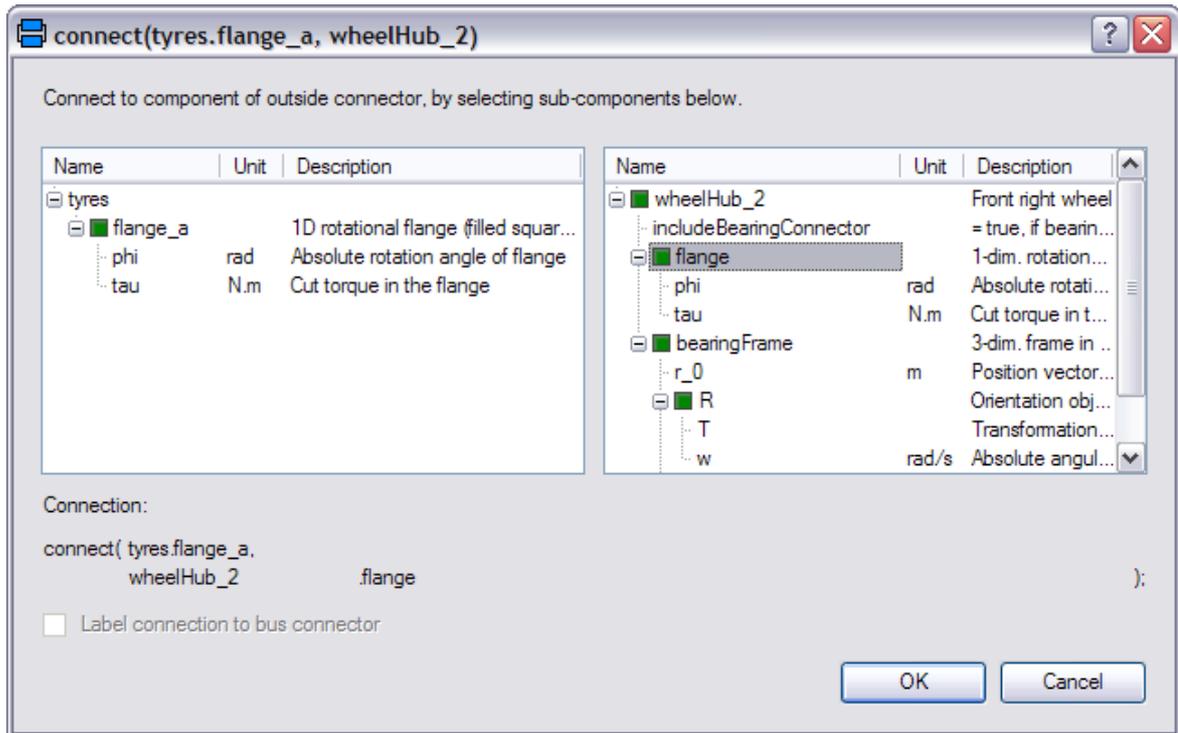
Starting from step 3 above.

1. First, decide which of the optional connectors are required in the model. For this example we don't need any of the optional connections

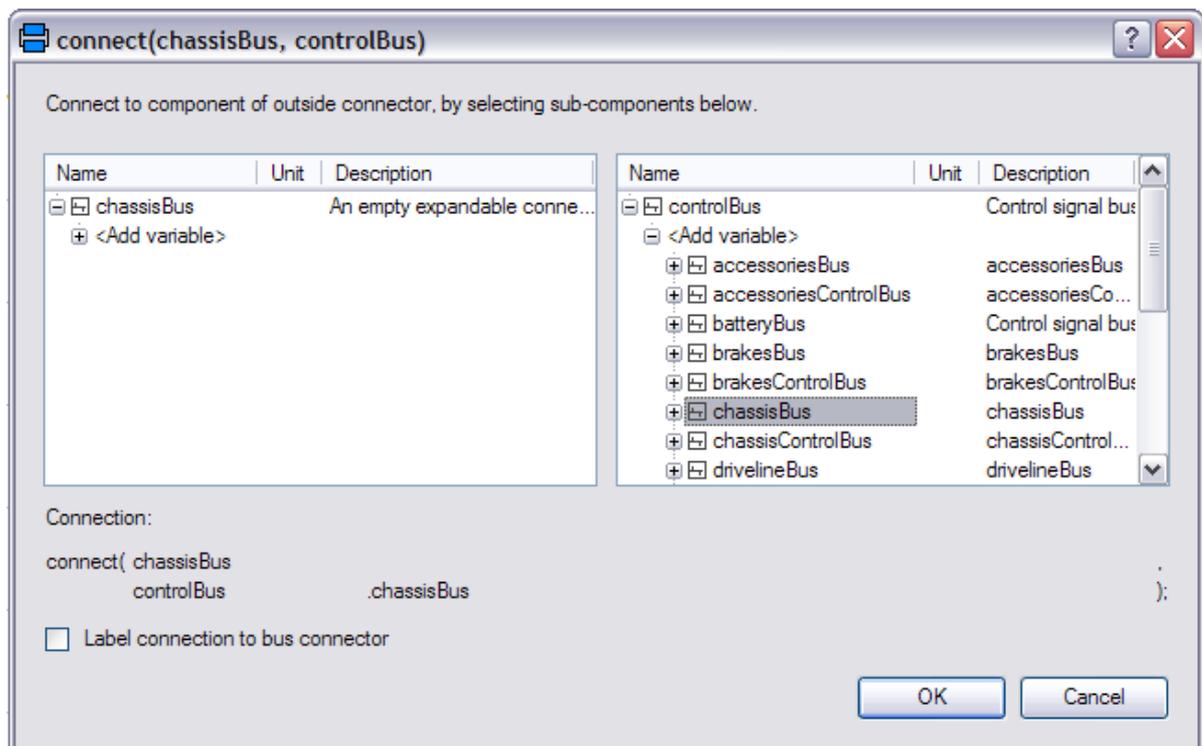


2. Add the following blocks and connections to the diagram. When you draw the connections to the wheelHub connectors the dialog box shown below will appear asking which connector within the wheelHub connector you would like to make the connection to. As we are modelling the wheels as a 1D system you should select **flange** from the list of options which is the 1D connector within the wheelHub connector.

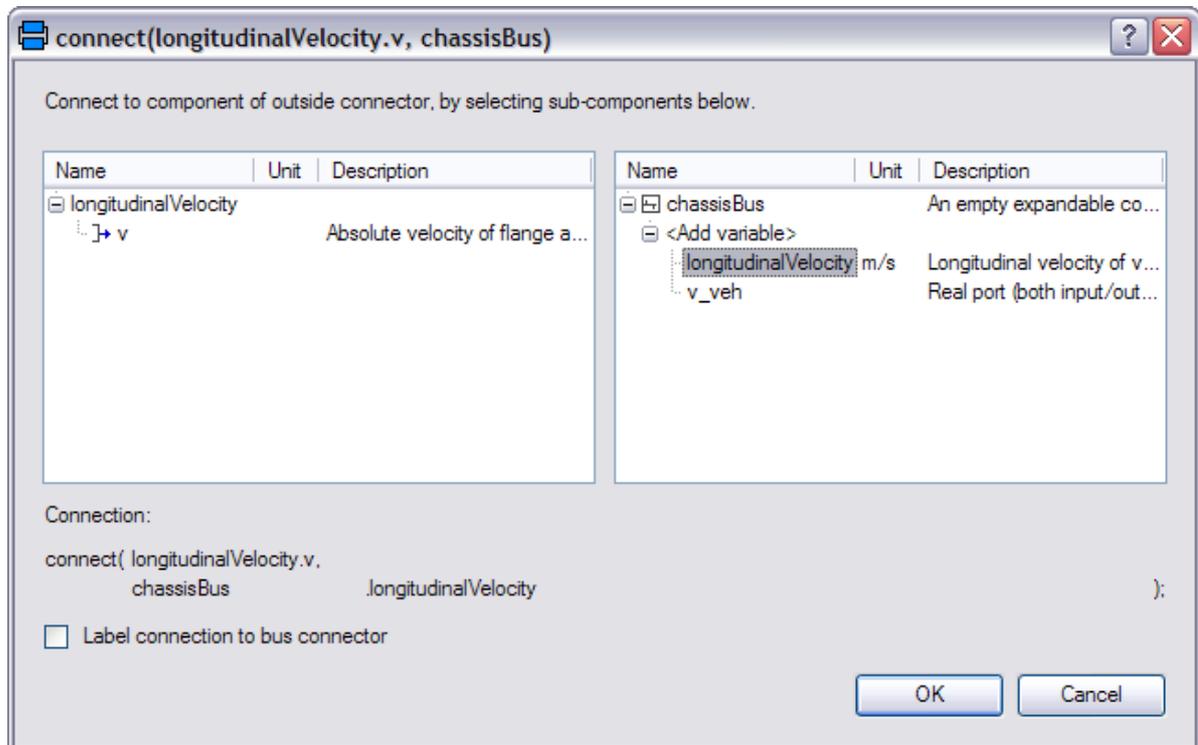




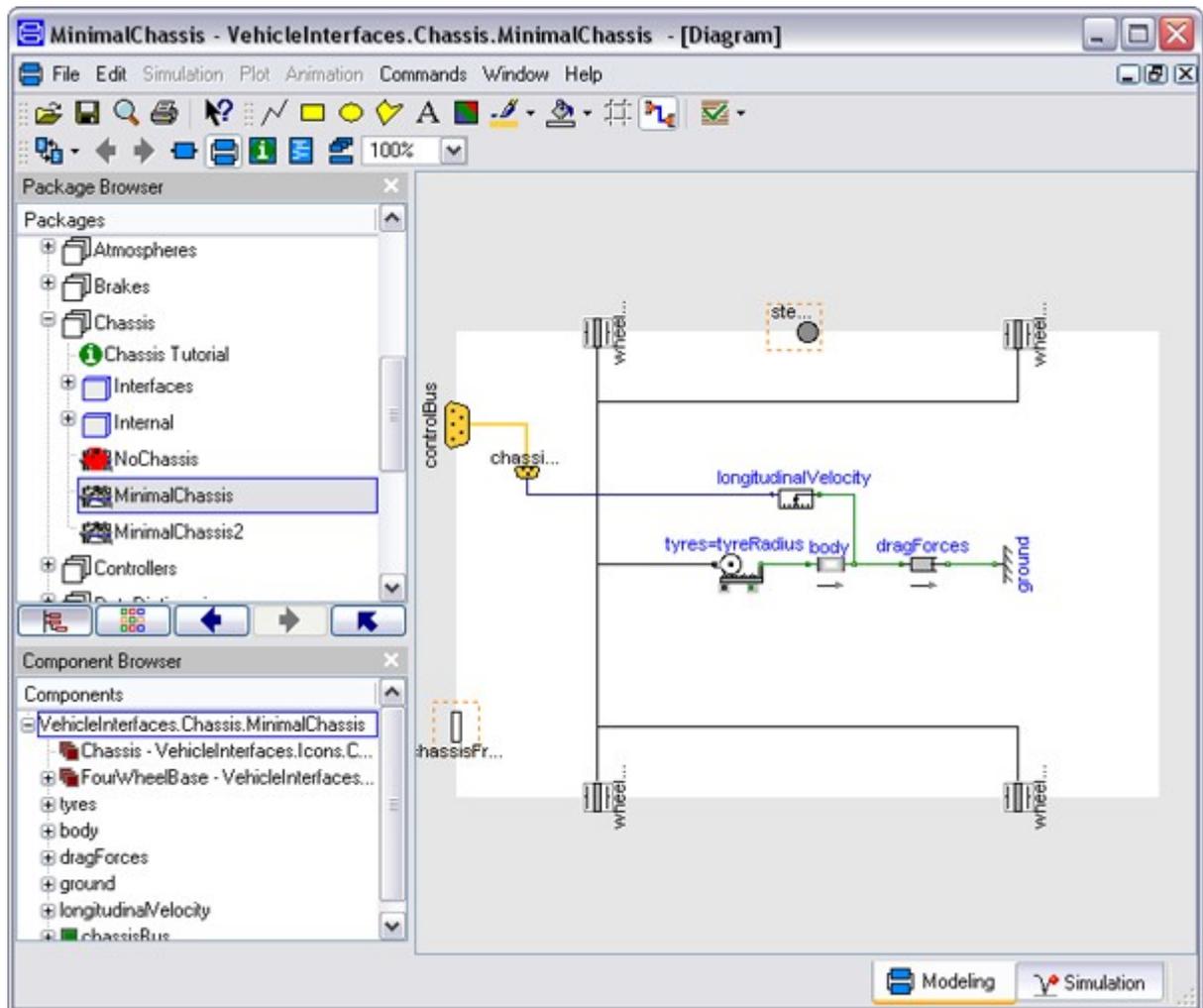
3. Next, we need to check to see if any connections to the control signal bus are required for the chassis, see [here](#) for a complete list of the minimum connections required. In this case we need to add the vehicle longitudinal velocity to the control signal bus and this can be done by connecting a speed sensor to the vehicle body and then connecting this to the signal bus. As the longitudinal speed signals are added to the chassisBus we first need to add this connector. The chassisBus connector is **VehicleInterfaces.Interfaces.ChassisBus**. This should be connected to the **controlBus**, when this connection is made the following dialog is produced and should be completed as shown.



- When the connection between the sensor and the chassisBus connector is added the dialog below appears and should be completed as shown



- The model is now complete and should check successfully and can be used in any model compatible with the VehicleInterfaces library assuming the selected Driver model doesn't use the steering wheel or chassis frame connections



VehicleInterfaces.Chassis.Interfaces

Collection of interface definitions for chassis

Information

A collection of partial base classes which define interfaces for chassis models.

Extends from Modelica.Icons.InterfacesPackage (Icon for packages containing interfaces).

Package Content

Name	Description
 Base	Basic interface definition for a chassis
 TwoAxleBase	Interface definition for a chassis of a 4 wheeled vehicle
 ThreeAxleBase	Interface definition for a chassis of a 6 wheeled vehicle
 FourAxleBase	Interface definition for a chassis of a 8 wheeled vehicle
 StandardBus	Bus of VehicleInterfaces.Chassis: StandardBus of signals generated by the chassis

VehicleInterfaces.Chassis.Interfaces.Base

Basic interface definition for a chassis



Information

This partial model defines the basic interfaces required for any chassis subsystem within the VehicleInterfaces package. This class should be extended to form a chassis interface definition with the correct number of wheelHub connectors for the type of vehicle being modelled. See the [documentation](#) and [tutorial](#) for more information.

Parameters

Name	Description
Advanced	
usingMultiBodyDriveline	=true if using a MultiBody driveline with a 1D chassis

Connectors

Name	Description
controlBus	Control signal bus
chassisFrame	Chassis reference frame (optional)
steeringWheel	Steering wheel connection (optional)

VehicleInterfaces.Chassis.Interfaces.TwoAxleBase

Interface definition for a chassis of a 4 wheeled vehicle



Information

This partial model defines the interfaces required for the chassis subsystem of a four wheeled vehicle within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Extends from [Base](#) (Basic interface definition for a chassis).

Parameters

Name	Description
Advanced	
usingMultiBodyDriveline	=true if using a MultiBody driveline with a 1D chassis

Connectors

Name	Description
controlBus	Control signal bus
chassisFrame	Chassis reference frame (optional)
steeringWheel	Steering wheel connection (optional)
wheelHub_1	Front left wheel
wheelHub_2	Front right wheel
wheelHub_3	Rear left wheel
wheelHub_4	Rear right wheel

VehicleInterfaces.Chassis.Interfaces.ThreeAxleBase

Interface definition for a chassis of a 6 wheeled vehicle

**Information**

This partial model defines the interfaces required for the chassis subsystem of a three axled vehicle within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Extends from [Base](#) (Basic interface definition for a chassis).

Parameters

Name	Description
Advanced	
usingMultiBodyDriveline	=true if using a MultiBody driveline with a 1D chassis

Connectors

Name	Description
controlBus	Control signal bus
chassisFrame	Chassis reference frame (optional)
steeringWheel	Steering wheel connection (optional)
wheelHub_1	Front left wheel
wheelHub_2	Front right wheel
wheelHub_3	Second axle left wheel
wheelHub_4	Second axle right wheel
wheelHub_5	Third axle left wheel
wheelHub_6	Third axle right wheel

VehicleInterfaces.Chassis.Interfaces.FourAxleBase

Interface definition for a chassis of a 8 wheeled vehicle

**Information**

This partial model defines the interfaces required for the chassis subsystem of a four axled vehicle within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Extends from [Base](#) (Basic interface definition for a chassis).

Parameters

Name	Description
Advanced	
usingMultiBodyDriveline	=true if using a MultiBody driveline with a 1D chassis

Connectors

Name	Description
controlBus	Control signal bus
chassisFrame	Chassis reference frame (optional)

steeringWheel	Steering wheel connection (optional)
wheelHub_1	Front left wheel
wheelHub_2	Front right wheel
wheelHub_3	Second axle left wheel
wheelHub_4	Second axle right wheel
wheelHub_5	Third axle left wheel
wheelHub_6	Third axle right wheel
wheelHub_7	Fourth axle left wheel
wheelHub_8	Fourth axle right wheel

VehicleInterfaces.Chassis.Interfaces.StandardBus

Bus of VehicleInterfaces.Chassis: StandardBus of signals generated by the chassis



Information

Bus with a set of standard signals generated by the chassis subsystem.

Extends from [VehicleInterfaces.Interfaces.ChassisBus](#) (Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as chassis bus), [VehicleInterfaces.Icons.SignalSubBusWithExplicitSignals](#) (Icon for signal sub-bus where the explicit signals are defined in the bus).

Parameters

Name	Description
longitudinalVelocity	Longitudinal velocity of vehicle [m/s]

Contents

Name	Description
longitudinalVelocity	Longitudinal velocity of vehicle [m/s]

VehicleInterfaces.Chassis.NoChassis

Empty chassis



Information

Empty chassis model. Using this empty model in overall vehicle architecture the functionality of chassis can be eliminated.

Extends from [VehicleInterfaces.Icons.Chassis](#) (Icon for a chassis subsystem), [VehicleInterfaces.Icons.Empty](#) (Icon for an empty component), [VehicleInterfaces.Chassis.Interfaces.TwoAxleBase](#) (Interface definition for a chassis of a 4 wheeled vehicle).

Parameters

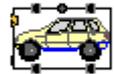
Name	Description
Advanced	
usingMultiBodyDriveline	=true if using a MultiBody driveline with a 1D chassis

Connectors

Name	Description
controlBus	Control signal bus
chassisFrame	Chassis reference frame (optional)
steeringWheel	Steering wheel connection (optional)
wheelHub_1	Front left wheel
wheelHub_2	Front right wheel
wheelHub_3	Rear left wheel
wheelHub_4	Rear right wheel

VehicleInterfaces.Chassis.MinimalChassis

Basic chassis model with rigid connection between all 4 wheels

**Information**

Single degree-of-freedom chassis model with mass and speed dependant drag model

Extends from [VehicleInterfaces.Icons.Chassis](#) (Icon for a chassis subsystem),

[VehicleInterfaces.Chassis.Interfaces.TwoAxleBase](#) (Interface definition for a chassis of a 4 wheeled vehicle).

Parameters

Name	Description
vehicleMass	Vehicle mass [kg]
tyreRadius	Tyre rolling radius [m]
drag	Drag force (force = drag*vehicle_speed)
Advanced	
usingMultiBodyDriveline	=true if using a MultiBody driveline with a 1D chassis

Connectors

Name	Description
controlBus	Control signal bus
chassisFrame	Chassis reference frame (optional)
steeringWheel	Steering wheel connection (optional)
wheelHub_1	Front left wheel
wheelHub_2	Front right wheel
wheelHub_3	Rear left wheel
wheelHub_4	Rear right wheel

VehicleInterfaces.Chassis.MinimalChassis2

Basic chassis model with rigid connection between all 4 wheels, uses chassisFrame connection

**Information**

Single degree-of-freedom chassis model with mass and speed dependant drag model. Couples the 1d

system to a Multibody system to support the other subsystems.

Extends from [VehicleInterfaces.Icons.Chassis](#) (Icon for a chassis subsystem),

[VehicleInterfaces.Chassis.Interfaces.TwoAxleBase](#) (Interface definition for a chassis of a 4 wheeled vehicle).

Parameters

Name	Description
vehicleMass	Vehicle mass [kg]
tyreRadius	Tyre rolling radius [m]
drag	Drag force (force = drag*vehicle_speed)
Advanced	
usingMultiBodyDriveline	=true if using a MultiBody driveline with a 1D chassis

Connectors

Name	Description
controlBus	Control signal bus
chassisFrame	Chassis reference frame (optional)
steeringWheel	Steering wheel connection (optional)
wheelHub_1	Front left wheel
wheelHub_2	Front right wheel
wheelHub_3	Rear left wheel
wheelHub_4	Rear right wheel

VehicleInterfaces.Chassis.MinimalChassis3

Basic chassis model with rigid connection between all 4 wheels, uses chassisFrame connection, includes wheel bearings



Information

Single degree-of-freedom chassis model with mass and speed dependant drag model. Uses MultiBody wheelHub connections and positions these relative to the vehicle body position.

Extends from [MinimalChassis2](#) (Basic chassis model with rigid connection between all 4 wheels, uses chassisFrame connection).

Parameters

Name	Description
vehicleMass	Vehicle mass [kg]
tyreRadius	Tyre rolling radius [m]
drag	Drag force (force = drag*vehicle_speed)
Advanced	
usingMultiBodyDriveline	=true if using a MultiBody driveline with a 1D chassis

Connectors

Name	Description
controlBus	Control signal bus

chassisFrame	Chassis reference frame (optional)
steeringWheel	Steering wheel connection (optional)
wheelHub_1	Front left wheel
wheelHub_2	Front right wheel
wheelHub_3	Rear left wheel
wheelHub_4	Rear right wheel

VehicleInterfaces.Controllers

Collection of controllers subsystem definitions

Information

The controller subsystem interface is defined in this sub-package of the VehicleInterfaces library. The controller subsystem has the following connector:

- **controlBus** - control signal bus connection

Effects to be modelled in this subsystem

Within the VehicleInterfaces package the controller subsystem is a generic controller interface definition and should be used to model the vehicle control systems in an appropriate manner.

Extends from [VehicleInterfaces.Icons.VariantLibrary](#) (Icon for a package class which contains several variants of one assembly or component).

Package Content

Name	Description
 Tutorial	Controllers Tutorial
 Interfaces	Collection of interface definitions for controller
 NoController	Empty controller

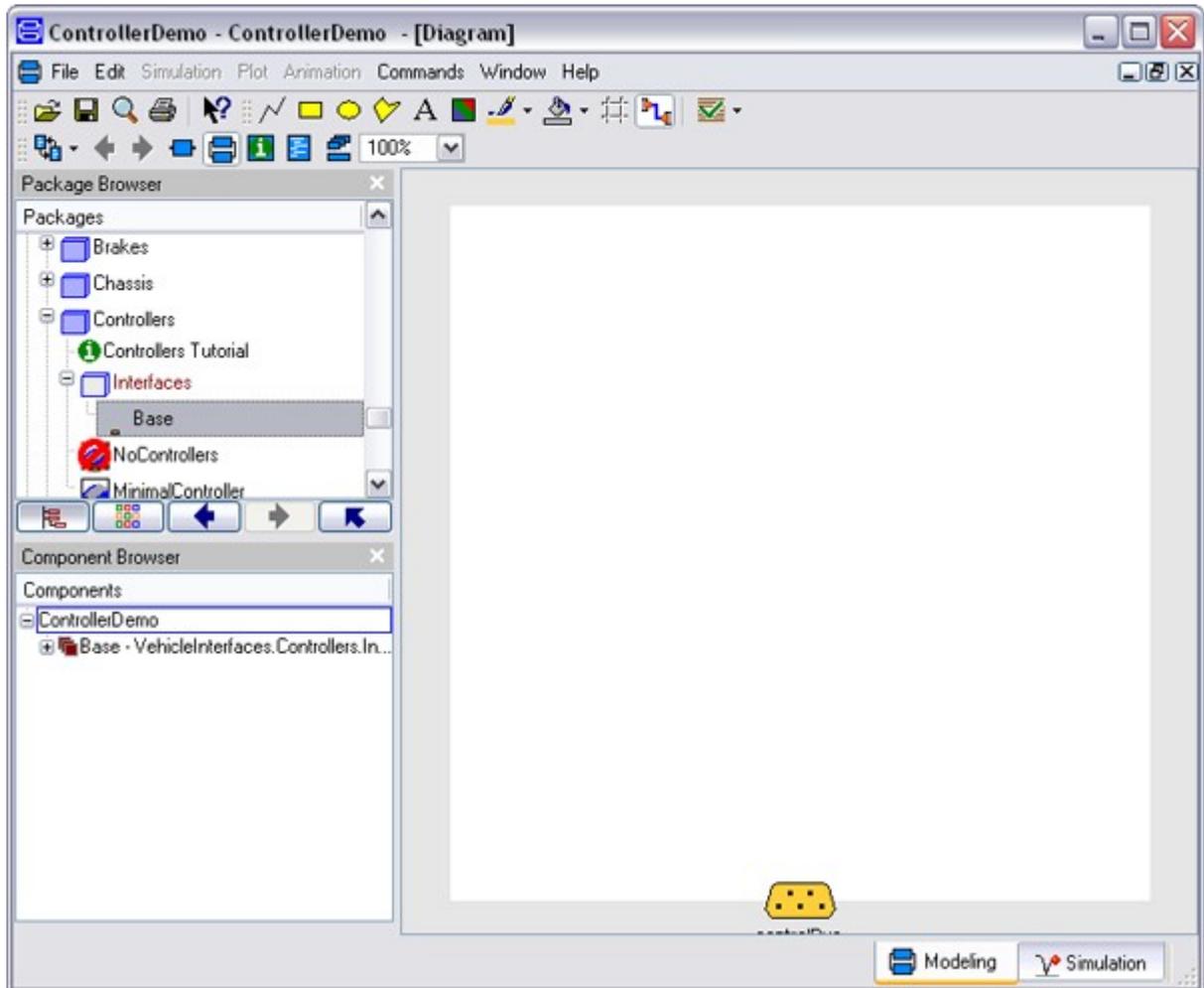
VehicleInterfaces.Controllers.Tutorial

Tutorial - Defining a new controller model

The following process will demonstrate how to create a new controller model using these interface definitions. This tutorial will guide you through building a simple idle speed controller for an engine.



1. Create a new model that extends **VehicleInterfaces.Controllers.Interfaces.Base**, it should look like this:



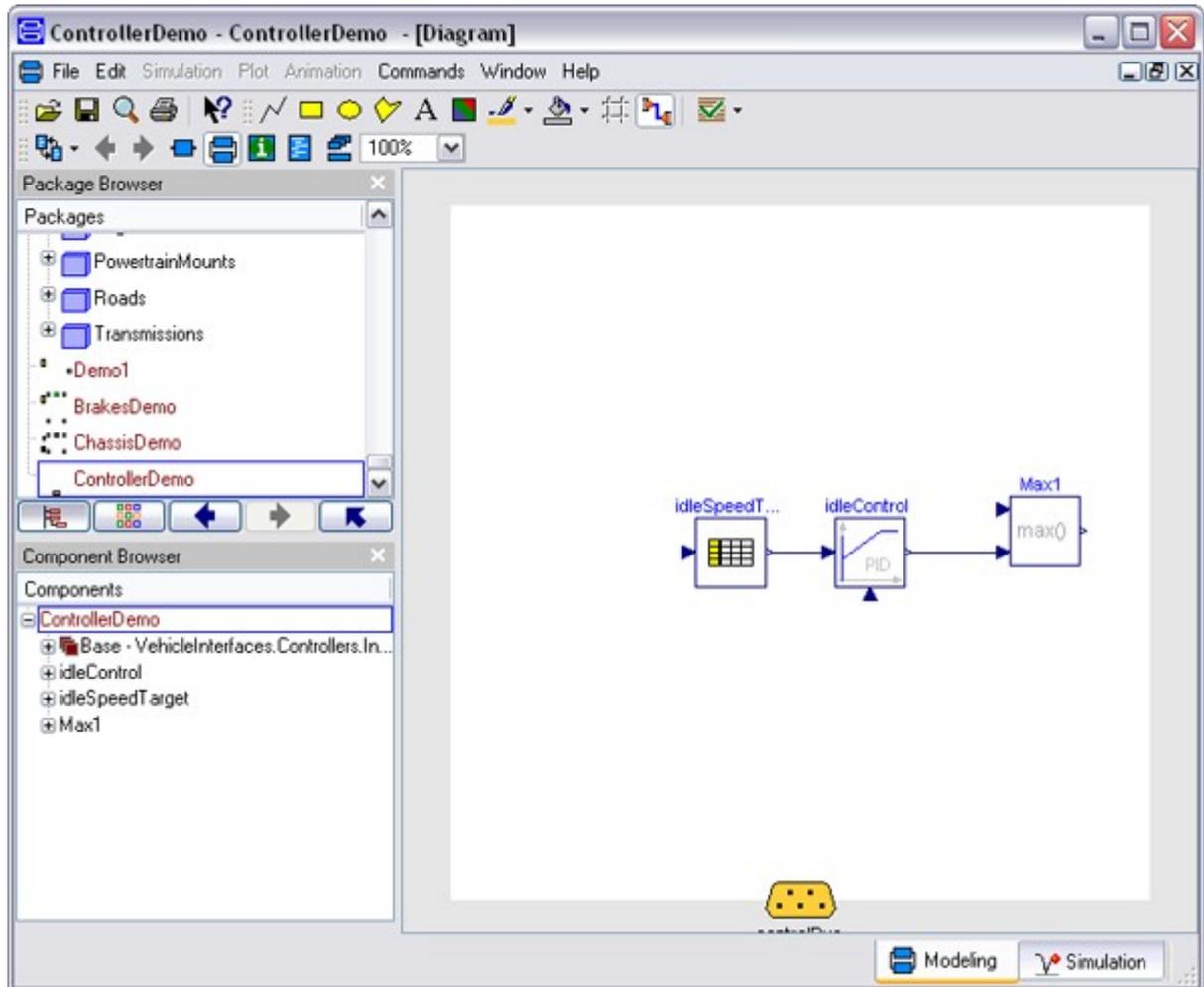
2. You can now define your controller model as required

Creating a simple idle speed controller example

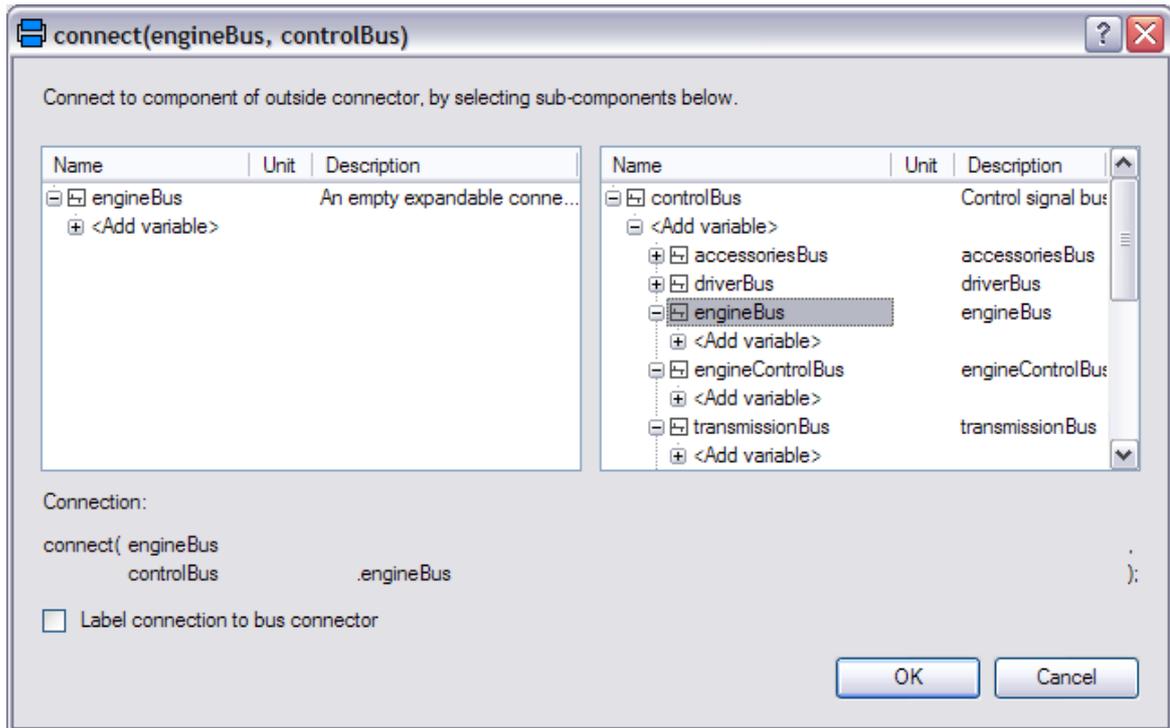
The following steps demonstrate how to create a simple idle speed controller model. The controller model will sense the engine speed, coolant temperature and accelerator pedal position and adjust the throttle signal to the engine to attempt to maintain the correct idle speed.

Starting from step 2 above.

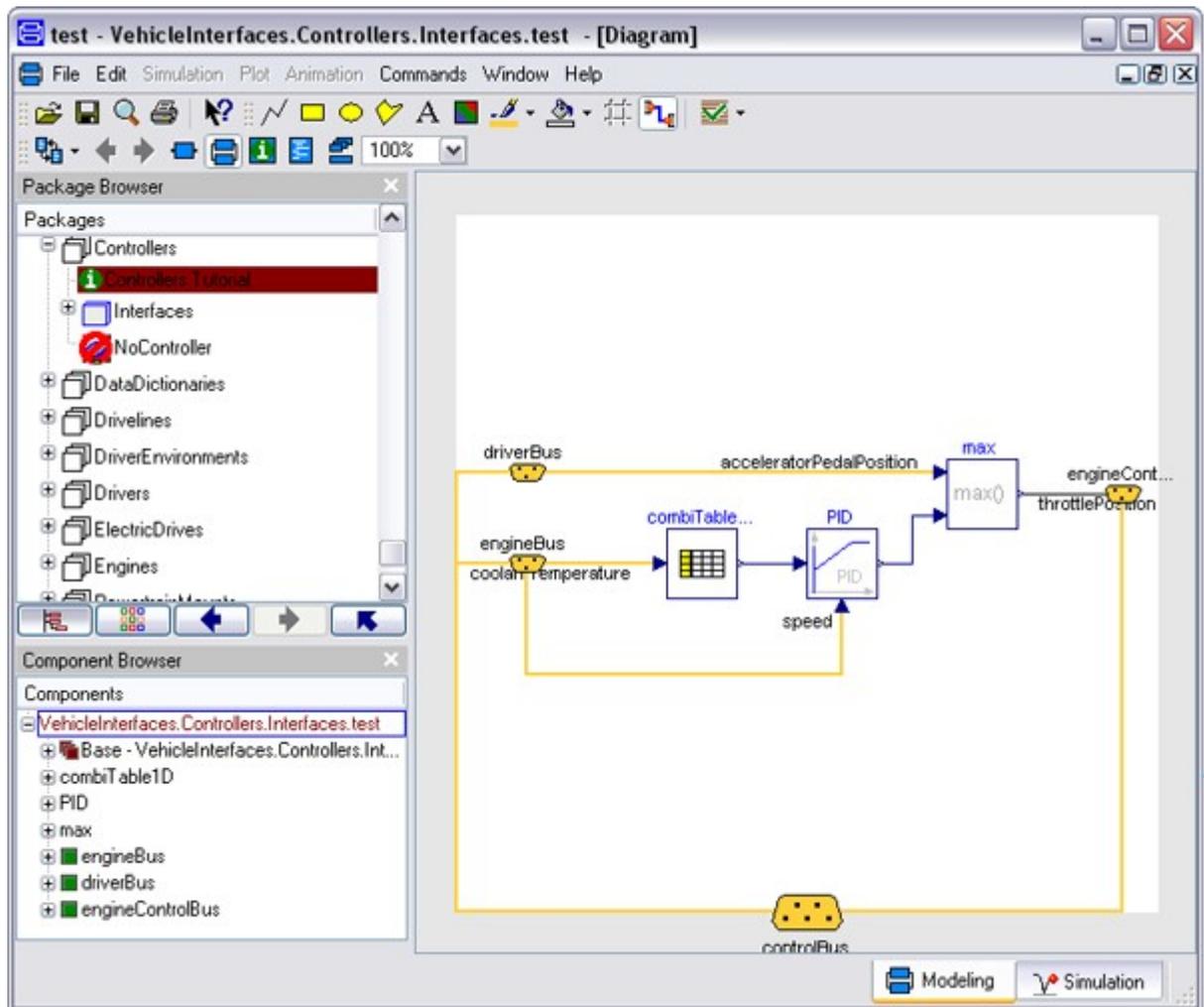
1. Add the following blocks and connections to the diagram.



2. The controller model is built on the assumption that a corresponding engine model will be developed that outputs the required signals to the control signal bus. We need to add connections to the control signal bus and will assume the following names for the additional signals required that are not part of the minimal set defined as part of the VehicleInterfaces
 - engineBus.coolantTemperature
 - engineBus.acceleratorPedalPosition
 - engineControlBus.throttlePosition
3. To add these control signals we first need to add the **engineBus** and **engineControlBus** connectors and connect them to the **controlBus**. These connectors are **VehicleInterfaces.Interfaces.EngineBus** and **VehicleInterfaces.Interfaces.EngineControlBus**. When connecting these to the controlBus the following dialog will be produced and should be completed as following substituting **engineControlBus** for **engineBus** as appropriate.



4. We can then complete the model by connecting the blocks to the control bus connectors and using the appropriate signal names. The finished model should look like:



- The model is now complete and should check successfully and can be used in any model compatible with the VehicleInterfaces library assuming the selected engine model puts the required signals on to the control signal bus

VehicleInterfaces.Controllers.Interfaces

Collection of interface definitions for controller

Information

A collection of partial base classes which define interfaces for control system models.

Extends from Modelica.Icons.InterfacesPackage (Icon for packages containing interfaces).

Package Content

Name	Description
Base	Basic interface definition for a controller

VehicleInterfaces.Controllers.Interfaces.Base

Basic interface definition for a controller

Information

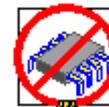
This partial model defines the interfaces required for a controller subsystem within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Connectors

Name	Description
controlBus	Control signal bus

VehicleInterfaces.Controllers.NoController

Empty controller



Information

Empty controller model. Using this empty model in overall vehicle architecture the functionality of controller can be eliminated.

Extends from [VehicleInterfaces.Icons.Controller](#) (Icon for a controller subsystem), [VehicleInterfaces.Icons.Empty](#) (Icon for an empty component), [Interfaces.Base](#) (Basic interface definition for a controller).

Connectors

Name	Description
controlBus	Control signal bus

VehicleInterfaces.DataDictionaries

Collection of data dictionary definitions

Information

The data dictionary interface is defined in this sub-package of the VehicleInterfaces library. The data dictionary subsystem has the following connector:

- **controlBus** - control signal bus connection

Effects to be modelled in this subsystem

Within the VehicleIntefaces package the data dictionary subsystem is used to provide compatibility between the naming convention used in the VehicleInterfaces library and in-house naming conventions. This is achieved by providing alias names for the various signals on the control signal bus.

Extends from [VehicleInterfaces.Icons.VariantLibrary](#) (Icon for a package class which contains several variants of one assembly or component).

Package Content

Name	Description
 Tutorial	Data Dictionary Tutorial
 Interfaces	Collection of interface definitions for data dictionary
 NoDataDictionary	Empty data dictionary

 MinimalExample	Basic data dictionary example
--	-------------------------------

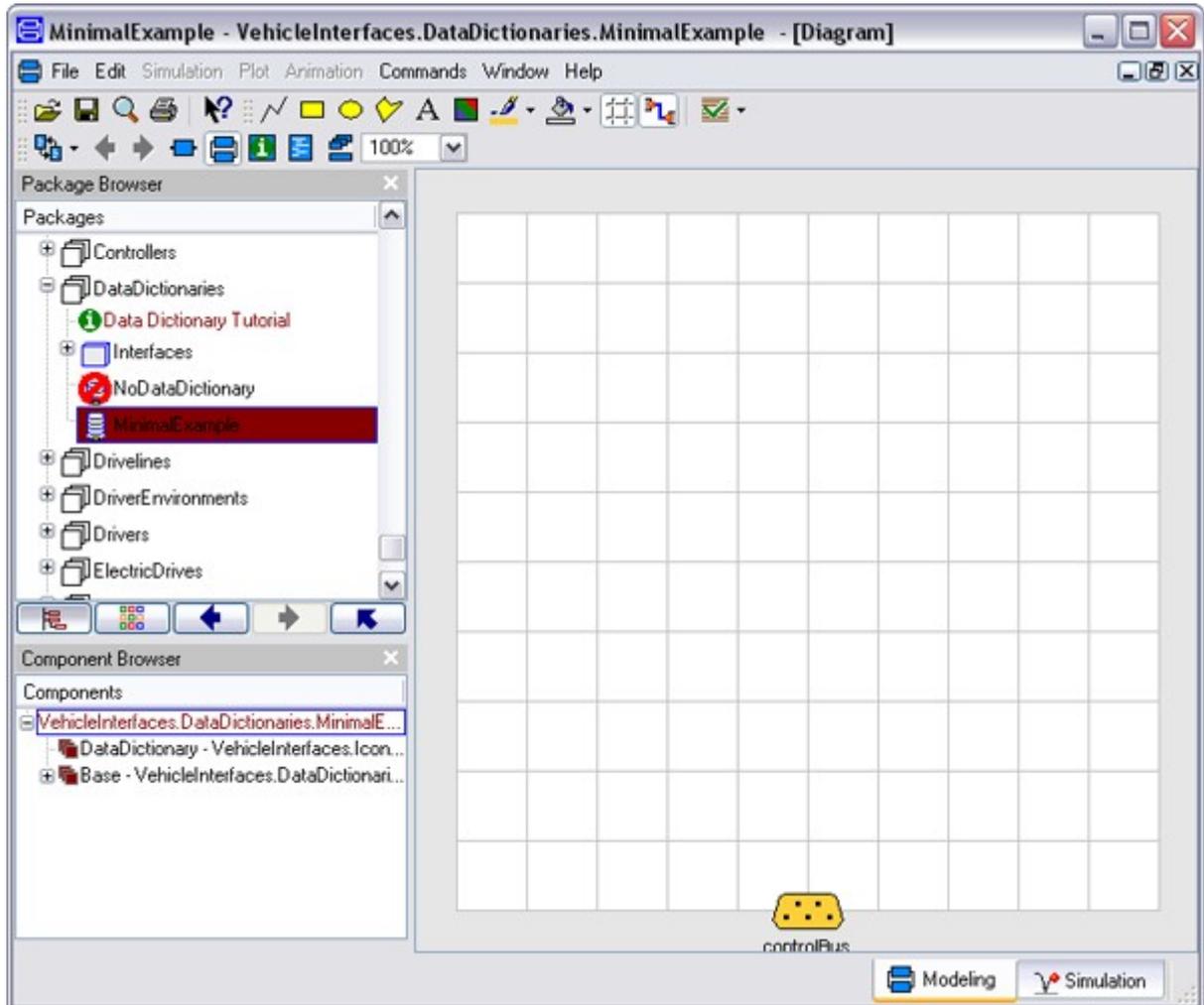
VehicleInterfaces.DataDictionaries.Tutorial

Tutorial - Defining a new data dictionary subsystem

The following process will demonstrate how to create a new data dictionary model using these interface definitions.



1. Create a new model that extends **VehicleInterfaces.DataDictionaries.Interfaces.Base**, it should look like this:



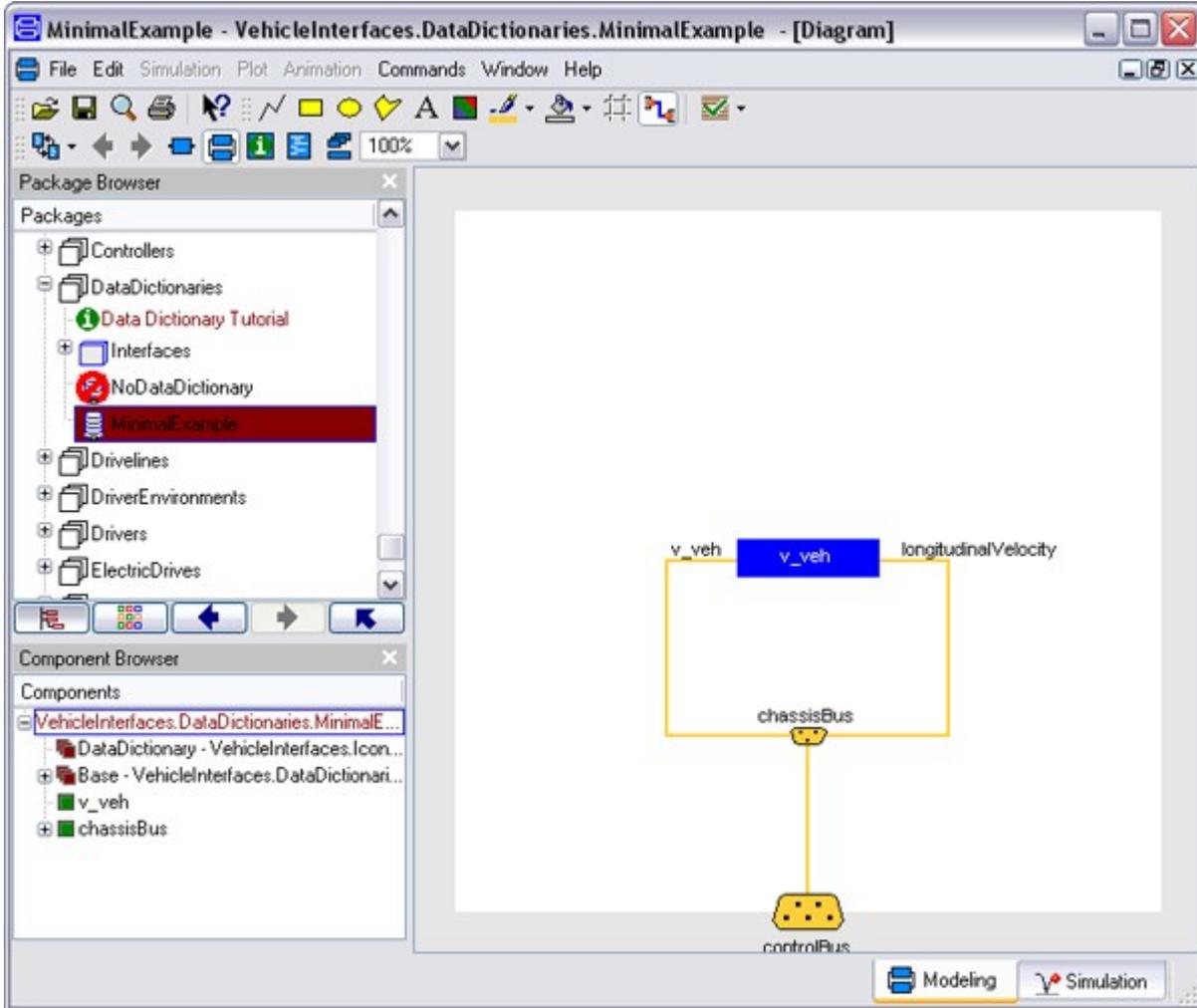
2. You can now define your data dictionary model as required

Adding a alias signal to the data dictionary

The following steps demonstrate how to add a new alias signal to the control signal bus. An alias signal for the vehicle longitudinal velocity in the chassisBus will be added called v_veh.

1. Add a block **Modelica.Blocks.Routing.RealPassThrough** and give it the name of the alias signal that you would like to create, in this case **v_veh**. Note, this block passes Real signal through without modification.
2. Then add the bus connector for the appropriate sub-bus, in this case the chassisBus connector which can be found at **VehicleInterfaces.Interfaces.ChassisBus**.
3. Then add two connections between the RealSignal connector and the chassisBus connector. The

first connects to the **longitudinalVelocity** signal within the chassisBus, the second creates a new signal called **v_veh**



VehicleInterfaces.DataDictionaries.Interfaces

Collection of interface definitions for data dictionary

Information

A collection of partial base classes which define interfaces for data dictionary.
 Extends from Modelica.Icons.InterfacesPackage (Icon for packages containing interfaces).

Package Content

Name	Description
Base	Basic interface definition for a data dictionary

VehicleInterfaces.DataDictionaries.Interfaces.Base

Basic interface definition for a data dictionary



Information

This partial model defines the interfaces required for a data dictionary within the VehicleInterfaces package. Especially, it provide alias names for the controBus. See the [documentation](#) and [tutorial](#) for more information.

Connectors

Name	Description
controlBus	Control signal bus

VehicleInterfaces.DataDictionaries.NoDataDictionary

Empty data dictionary



Information

Empty data dictionary model. Using this empty model in overall vehicle architecture the functionality of data dictionary can be eliminated.

Extends from [VehicleInterfaces.Icons.DataDictionary](#) (Icon for a data dictionary),
[VehicleInterfaces.Icons.Empty](#) (Icon for an empty component),
[VehicleInterfaces.DataDictionaries.Interfaces.Base](#) (Basic interface definition for a data dictionary).

Connectors

Name	Description
controlBus	Control signal bus

VehicleInterfaces.DataDictionaries.MinimalExample

Basic data dictionary example



Information

A simple data dictionary example that creates a new signal called **v_veh** within the **chassisBus** sub-bus of the control bus.

Extends from [VehicleInterfaces.Icons.DataDictionary](#) (Icon for a data dictionary),
[VehicleInterfaces.DataDictionaries.Interfaces.Base](#) (Basic interface definition for a data dictionary).

Connectors

Name	Description
controlBus	Control signal bus

VehicleInterfaces.Drivelines

Collection of drivelines subsystem definitions

Information

The driveline subsystem interfaces are defined in this sub-package of the VehicleInterfaces library. The driveline subsystem has the following connectors some of which are optional (see below for more information):

- **transmissionFlange** - 1D rotational connection to the transmission subsystem (or other systems)

- connected to the driveline input)
- **controlBus** - control signal bus connection
- **wheelHub_n** - wheelHub connectors that consist of a 1D rotational connector and a MultiBody frame connector (see [here](#). The number of these varies depending on how many wheels the vehicle has.
- **drivelineMount** - MultiBody connection to transmit the driveline mount reactions (optional)

The optional connectors are, by default, disabled and can be ignored if not required. They can be enabled by setting the appropriate parameter to be true. This is only possible at design time, i.e. when you are building the subsystem model.

Effects to be modelled in this subsystem

Within the VehicleInterfaces package the driveline subsystem is used to model the transmission of torque from the transmission output shaft to the wheel hubs. The connection to the transmission subsystem is a 1D rotational connectors. Different interface definitions are provided for vehicles with different numbers of wheels, a FlangeWithBearing connector is added for each wheel. The torque reaction in to the driveline housings and the housings themselves are also to be modelled in this subsystem if required. The torque reactions, if included, should all be referred back to a single reference frame (the drivelineMount connector).

Extends from [VehicleInterfaces.Icons.VariantLibrary](#) (Icon for a package class which contains several variants of one assembly or component).

Package Content

Name	Description
 Tutorial	Driveline Tutorial
 Interfaces	Collection of interface definitions for driveline
 NoDriveline	Empty driveline model for a 4 wheeled vehicle
 MinimalDriveline	Front wheel drive, 4 wheeled vehicle

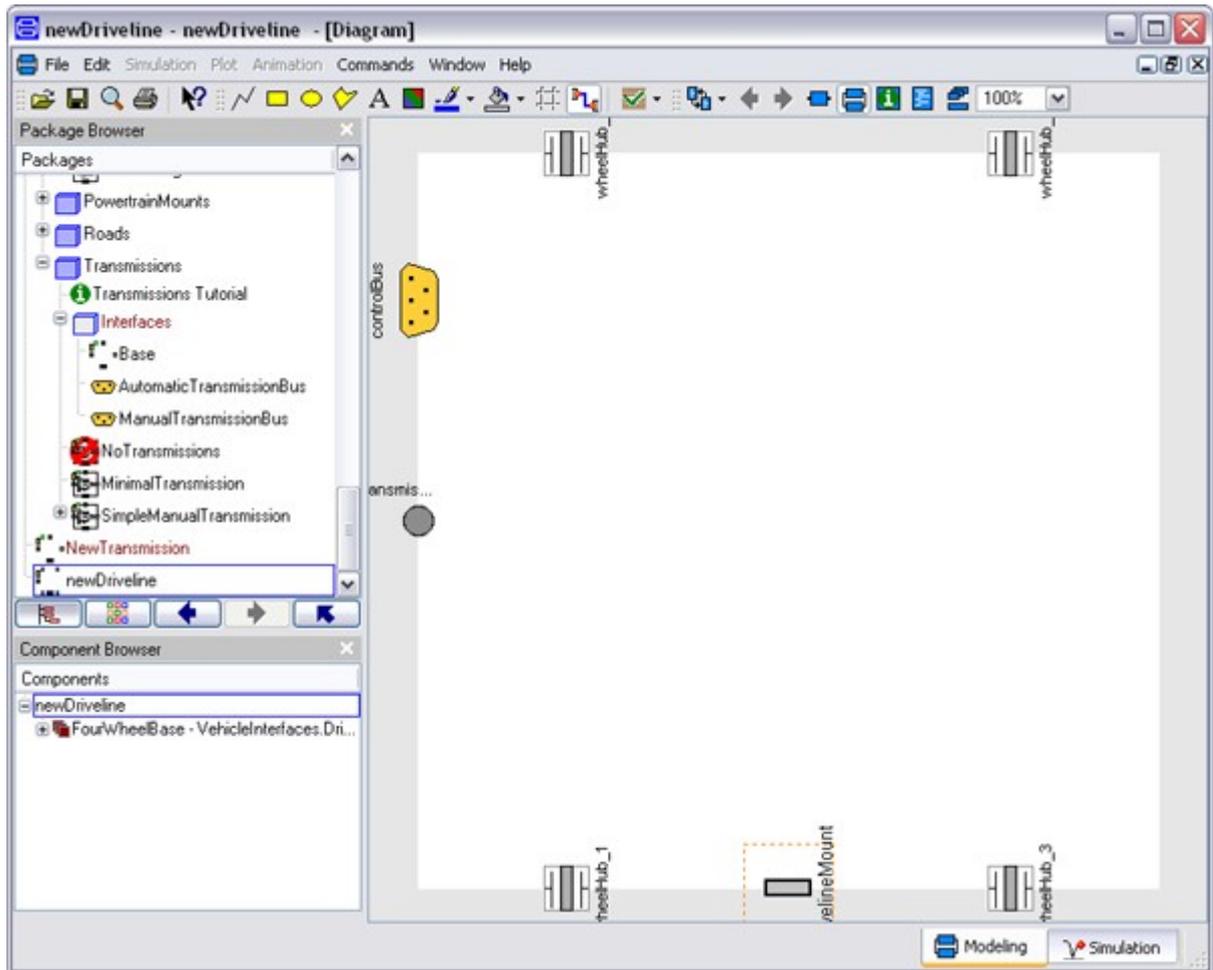
VehicleInterfaces.Drivelines.Tutorial

Tutorial - Defining a new driveline model

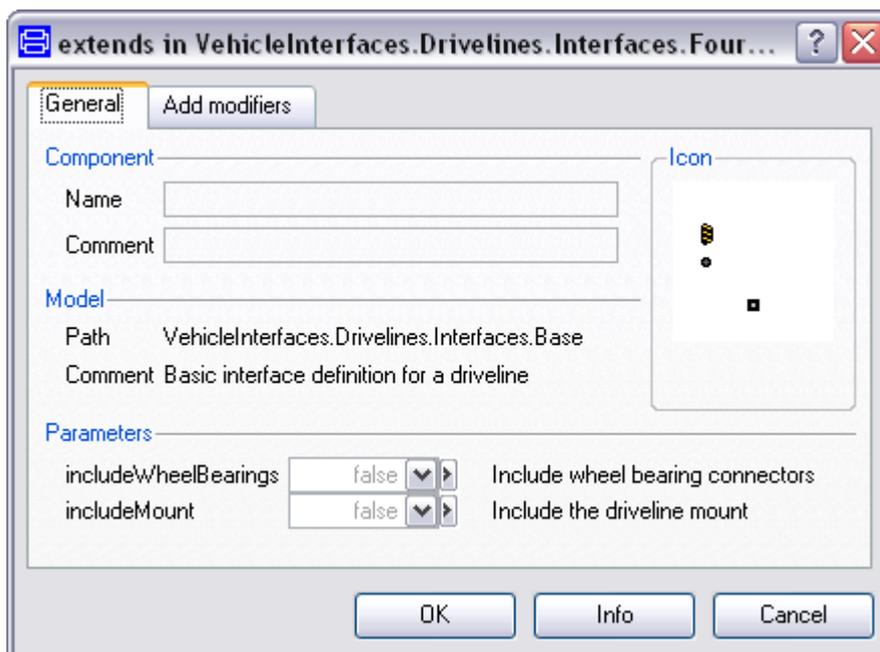
The following process will demonstrate how to create a new driveline model using these interface definitions. This tutorial will guide you through building a driveline for a passenger car, i.e. a vehicle with 4 wheels.



1. Create a new model that extends **VehicleInterfaces.Drivelines.Interfaces.TwoAxleBase**, it should look like this:



2. In the component browser, right click on **Base** and select **Parameters** from the context menu to produce the following parameter dialog



3. This dialog allows you to enable/disable the optional connections by setting **includeWheelBearings**

and **includeMount** as required for your new driveline model. The wheelHub connectors are of the type [VehicleInterfaces.Interfaces.FlangeWithBearing](#), the parameter **includeWheelBearings** controls whether the bearing connectors within the wheelHubs is enabled or not.

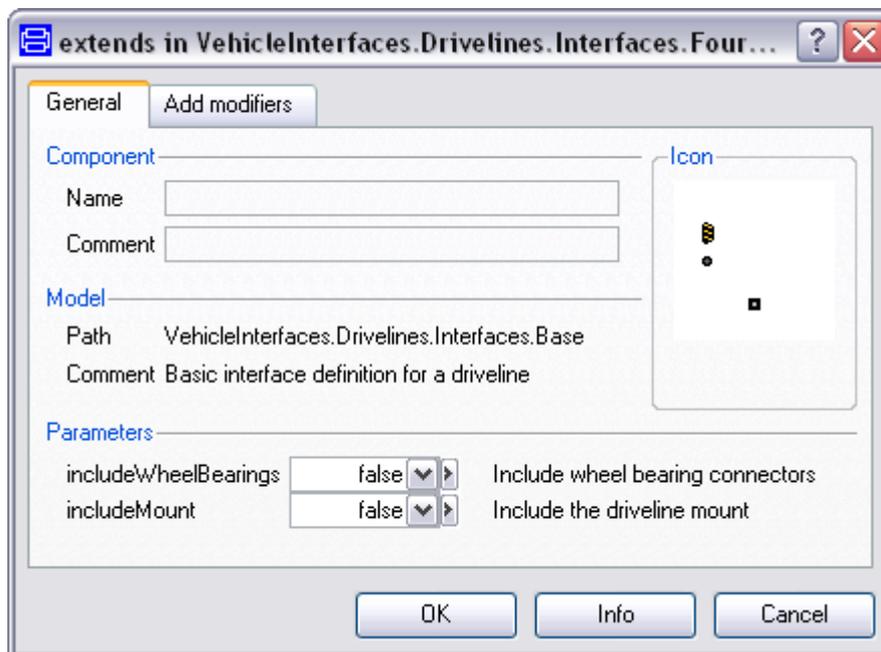
4. You can now define your driveline model as required

Creating a simple rear-wheel drive example

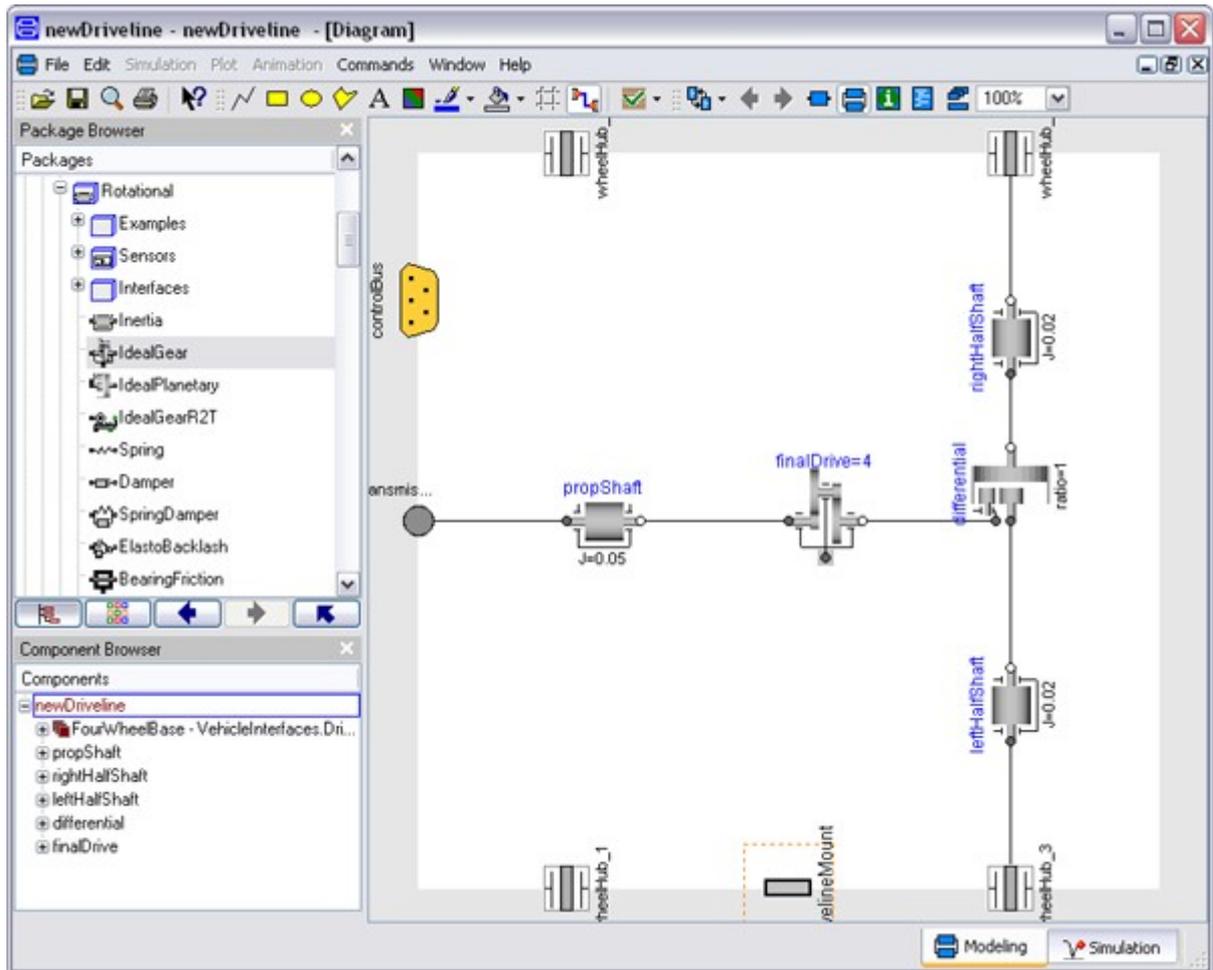
The following steps demonstrate how to create a simple rear-wheel drive driveline model. The driveline model will transmit the torque from the transmission to the rear wheels via a propshaft, differential with final drive and then two halfshafts. No torque reaction in to the transmission housings will be modelled.

Starting from step 3 above.

1. First, decide which of the optional connectors are required in the model. For this example we don't need any of the optional connections



2. Add the following blocks and connections to the diagram. When you draw the connections from the rightHalfShaft and leftHalfShaft components to the wheelHub connectors the dialog box shown below will appear asking which connector within the wheelHub connector you would like to make the connection to. As we are modelling the driveline as a 1D system you should select **flange** from the list of options which is the 1D connector within the wheelHub connector.



connect(tyres.flange_a, wheelHub_2)

Connect to component of outside connector, by selecting sub-components below.

Name	Unit	Description
tyres		
flange_a		1D rotational flange (filled squar...
phi	rad	Absolute rotation angle of flange
tau	N.m	Cut torque in the flange

Name	Unit	Description
wheelHub_2		Front right wheel
includeBearingConnector		= true, if bearin...
flange		1-dim. rotation...
phi	rad	Absolute rotati...
tau	N.m	Cut torque in t...
bearingFrame		3-dim. frame in ...
r_0	m	Position vector...
R		Orientation obj...
T		Transformation...
w	rad/s	Absolute angul...

Connection:

```
connect( tyres.flange_a,
        wheelHub_2      flange
        );
```

Label connection to bus connector

OK Cancel

3. Next, we need to check to see if any connections to the control signal bus are required for the driveline, see [here](#) for a complete list of the minimum connections required. In this case we don't need to add any signals to the control signal bus.
4. The model is now complete and should check successfully and can be used in any model compatible with the VehicleInterfaces library assuming the selected Driver model also uses the accelerator pedal connection

VehicleInterfaces.Drivelines.Interfaces

Collection of interface definitions for driveline

Information

A collection of partial base classes which define interfaces for driveline models.

Extends from Modelica.Icons.InterfacesPackage (Icon for packages containing interfaces).

Package Content

Name	Description
 Base	Basic interface definition for a driveline
 TwoAxleBase	Interface definition for a driveline of a 4 wheeled vehicle
 ThreeAxleBase	Interface definition for a driveline of a 6 wheeled vehicle
 FourAxleBase	Interface definition for a driveline of a 8 wheeled vehicle

VehicleInterfaces.Drivelines.Interfaces.Base

Basic interface definition for a driveline

Information

This partial model defines the basic interfaces required for any driveline subsystem within the VehicleInterfaces package. This class should be extended to form a driveline interface definition with the correct number of wheelHub connectors for the type of vehicle being modelled. See the [documentation](#) and [tutorial](#) for more information.

Parameters

Name	Description
Advanced	
usingMultiBodyChassis	=true if using a MultiBody chassis with a 1D driveline
usingMultiBodyTransmission	=true if using a MultiBody transmission with a 1D driveline

Connectors

Name	Description
transmissionFlange	Connection to the Transmission output shaft
controlBus	Control signal bus
drivelineMount	Driveline mount connection (optional)

VehicleInterfaces.Drivelines.Interfaces.TwoAxleBase

Interface definition for a driveline of a 4 wheeled vehicle

**Information**

This partial model defines the interfaces required for the driveline subsystem of a two axled vehicle within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Extends from [Base](#) (Basic interface definition for a driveline).

Parameters

Name	Description
Advanced	
usingMultiBodyChassis	=true if using a MultiBody chassis with a 1D driveline
usingMultiBodyTransmission	=true if using a MultiBody transmission with a 1D driveline

Connectors

Name	Description
transmissionFlange	Connection to the Transmission output shaft
controlBus	Control signal bus
drivelineMount	Driveline mount connection (optional)
wheelHub_1	Front left wheel
wheelHub_2	Front right wheel
wheelHub_3	Rear left wheel
wheelHub_4	Rear right wheel

VehicleInterfaces.Drivelines.Interfaces.ThreeAxleBase

Interface definition for a driveline of a 6 wheeled vehicle

**Information**

This partial model defines the interfaces required for the driveline subsystem of a three axled vehicle within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Extends from [VehicleInterfaces.Drivelines.Interfaces.Base](#) (Basic interface definition for a driveline).

Parameters

Name	Description
Advanced	
usingMultiBodyChassis	=true if using a MultiBody chassis with a 1D driveline
usingMultiBodyTransmission	=true if using a MultiBody transmission with a 1D driveline

Connectors

Name	Description
transmissionFlange	Connection to the Transmission output shaft
controlBus	Control signal bus

drivelineMount	Driveline mount connection (optional)
wheelHub_1	Front left wheel
wheelHub_2	Front right wheel
wheelHub_3	Second axle left wheel
wheelHub_4	Second axle right wheel
wheelHub_5	Third axle left wheel
wheelHub_6	Third axle right wheel

VehicleInterfaces.Drivelines.Interfaces.FourAxleBase

Interface definition for a driveline of a 8 wheeled vehicle



Information

This partial model defines the interfaces required for the driveline subsystem of a four axled vehicle within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Extends from [VehicleInterfaces.Drivelines.Interfaces.Base](#) (Basic interface definition for a driveline).

Parameters

Name	Description
Advanced	
usingMultiBodyChassis	=true if using a MultiBody chassis with a 1D driveline
usingMultiBodyTransmission	=true if using a MultiBody transmission with a 1D driveline

Connectors

Name	Description
transmissionFlange	Connection to the Transmission output shaft
controlBus	Control signal bus
drivelineMount	Driveline mount connection (optional)
wheelHub_1	Front left wheel
wheelHub_2	Front right wheel
wheelHub_3	Second axle left wheel
wheelHub_4	Second axle right wheel
wheelHub_5	Third axle left wheel
wheelHub_6	Third axle right wheel
wheelHub_7	Fourth axle left wheel
wheelHub_8	Fourth axle right wheel

VehicleInterfaces.Drivelines.NoDriveline

Empty driveline model for a 4 wheeled vehicle



Information

Zero torque is applied to all the wheelhubs and the reaction paths in to the wheel hubs included if the **driveTrainMechanics3D** flag in the world object is true.

Using this empty model in overall vehicle architecture the functionality of driveline can be eliminated.

Extends from [VehicleInterfaces.Icons.Driveline](#) (Icon for a driveline subsystem),
[VehicleInterfaces.Icons.Empty](#) (Icon for an empty component),
[VehicleInterfaces.Drivelines.Interfaces.TwoAxleBase](#) (Interface definition for a driveline of a 4 wheeled vehicle).

Parameters

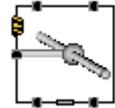
Name	Description
Advanced	
usingMultiBodyChassis	=true if using a MultiBody chassis with a 1D driveline
usingMultiBodyTransmission	=true if using a MultiBody transmission with a 1D driveline

Connectors

Name	Description
transmissionFlange	Connection to the Transmission output shaft
controlBus	Control signal bus
drivelineMount	Driveline mount connection (optional)
wheelHub_1	Front left wheel
wheelHub_2	Front right wheel
wheelHub_3	Rear left wheel
wheelHub_4	Rear right wheel

VehicleInterfaces.Drivelines.MinimalDriveline

Front wheel drive, 4 wheeled vehicle



Information

This driveline model is of a front-wheel drive 4-wheeled vehicle. The front differential is modelled using an ideal gear and planetary gear. The model does include the 3D mount effects if the **driveTrainMechanics3D** in the world object is set to true. To properly include these effects the additional constant and FrameForceAndTorque actuators are required on the front wheel hubs. A constant zero torque is applied to the rear wheelhubs and the reaction paths in to the wheelhubs are included.

Extends from [VehicleInterfaces.Icons.Driveline](#) (Icon for a driveline subsystem),
[VehicleInterfaces.Drivelines.Interfaces.TwoAxleBase](#) (Interface definition for a driveline of a 4 wheeled vehicle).

Parameters

Name	Description
halfshaftInertia	Inertia of each halfshaft [kg.m2]
finalDriveRatio	Final drive ratio
Advanced	
usingMultiBodyChassis	=true if using a MultiBody chassis with a 1D driveline
usingMultiBodyTransmission	=true if using a MultiBody transmission with a 1D driveline

Connectors

Name	Description
transmissionFlange	Connection to the Transmission output shaft
controlBus	Control signal bus
drivelineMount	Driveline mount connection (optional)
wheelHub_1	Front left wheel
wheelHub_2	Front right wheel
wheelHub_3	Rear left wheel
wheelHub_4	Rear right wheel

VehicleInterfaces.DriverEnvironments

Collection of driver environment subsystem definitions

Information

The driver environment subsystem interfaces are defined in this sub-package of the VehicleInterfaces library. The driver environment subsystem has the following connectors some of which are optional (see below for more information):

- **controlBus** - control signal bus connection
- **chassisFrame** - MultiBody connection providing a connection point to the vehicle body (optional)
- **steeringWheel** - steering wheel connection (optional);
- **acceleratorPedal** - 1D translational connection for the accelerator pedal connection to the driverEnvironment (optional)
- **brakePedal** - 1D translational connection for the brake pedal connection to the driverEnvironment (optional)
- **clutchPedal** - 1D translational connection for the clutch pedal connection to the driverEnvironment (optional, for manual gearboxes only)
- **shiftConnector** - shift connection to the driverEnvironment (optional, for manual gearboxes only)

The optional connectors are, by default, disabled and can be ignored if not required. They can be enabled by setting the appropriate parameter to be true. This is only possible at design time, i.e. when you are building the subsystem model.

Effects to be modelled in this subsystem

Within the VehicleInterfaces package the driver environment subsystem is used to model the interaction between the human driver and the vehicle itself. A library developer may also choose to include the driver logic within this subsystem or they may use a separate Driver subsystem based on VehicleInterfaces.Drivers.Interfaces.Base.

Extends from [VehicleInterfaces.Icons.VariantLibrary](#) (Icon for a package class which contains several variants of one assembly or component).

Package Content

Name	Description
 Tutorial	Driver Environment Tutorial
 Interfaces	Collection of interface definitions for driver environment
 NoDriverEnvironment	Empty driver-vehicle interface
 DriveByWireAutomatic	Minimal Drive-by-wire Driver-Vehicle Interface

 ConventionalManual	Minimal Driver-Vehicle Interface, manual transmission, uses physical pedal connections
 DriveByWireAutomaticExternalDriver	Minimal drive-by-wire driver-vehicle interface with external driver model

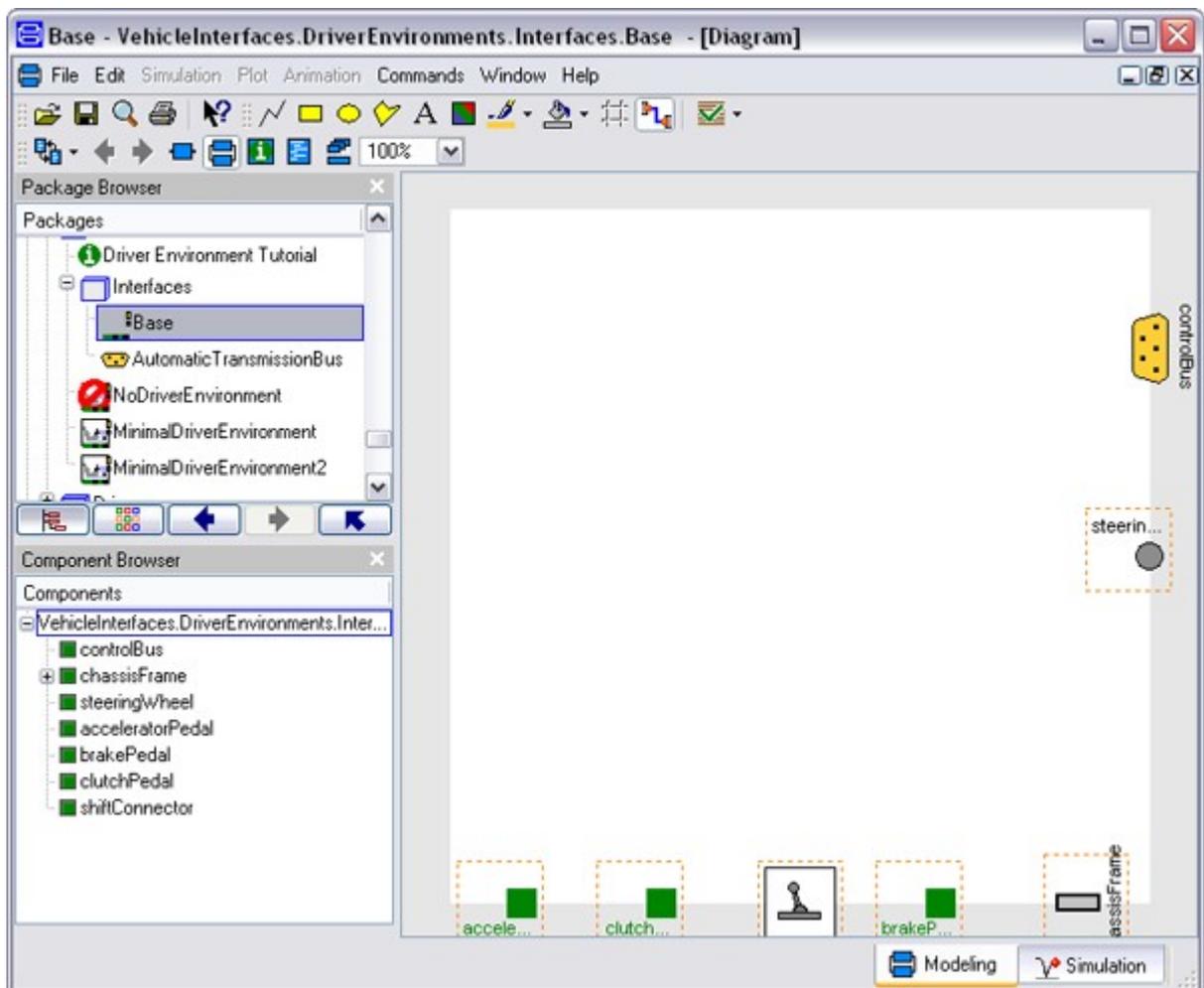
VehicleInterfaces.DriverEnvironments.Tutorial

Tutorial - Defining a new driver environment model

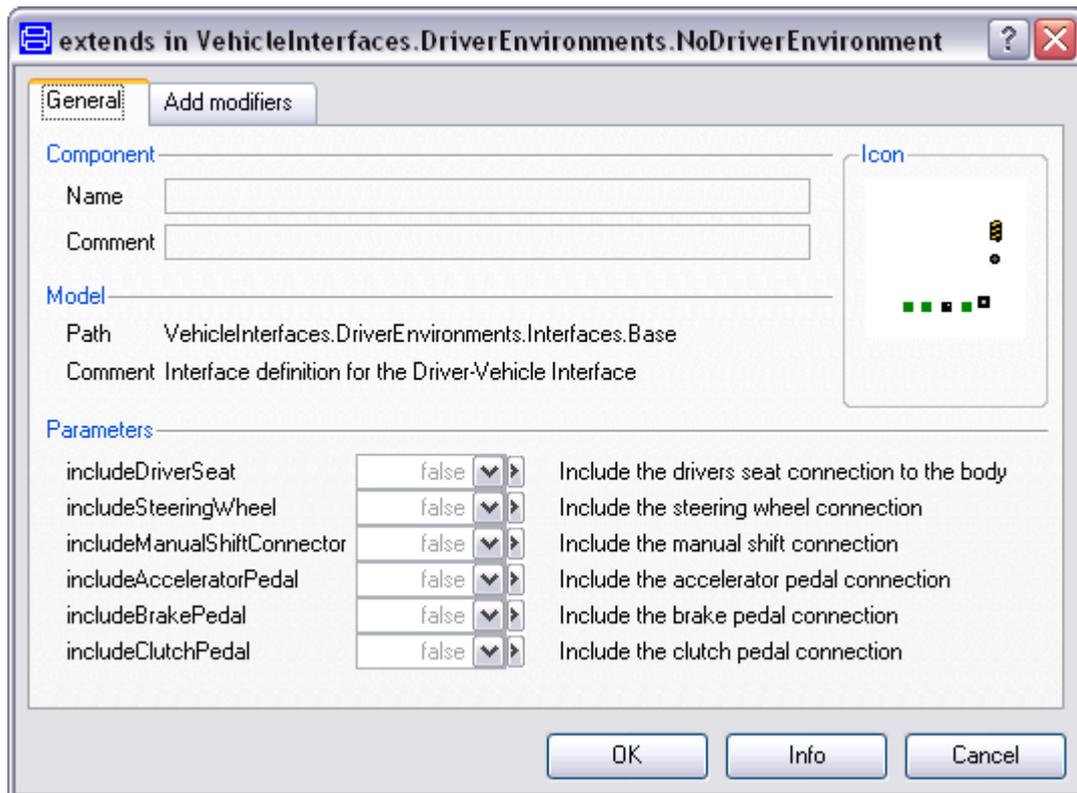
The following process will demonstrate how to create a new driver environment model using these interface definitions. This tutorial will guide you through building a driver environment for a conventional passenger car with a manual transmission.



1. Create a new model that extends **VehicleInterfaces.DriverEnvironments.Interfaces.BaseManualTransmission**, it should look like this:



2. In the component browser, right click on **Base** and select **Parameters** from the context menu to produce the following parameter dialog



3. This dialog allows you to enable/disable the optional connections by setting the various parameters as required for your new driver environment model.
4. You can now define your driver environment model as required

VehicleInterfaces.DriverEnvironments.Interfaces

Collection of interface definitions for driver environment

Information

A collection of partial base classes which define interfaces for driver environment models.
 Extends from Modelica.Icons.InterfacesPackage (Icon for packages containing interfaces).

Package Content

Name	Description
 Base	Basic interface definition for the driver-vehicle interface
 BaseAutomaticTransmission	Interface definition for the driver-vehicle interface with an automatic transmission
 BaseAutomaticTransmissionExternalDriver	Interface definition for the driver-vehicle interface with an automatic transmission and external driver
 BaseManualTransmission	Interface definition for the driver-vehicle interface with a manual transmission
 BaseManualTransmissionExternalDriver	Interface definition for the driver-vehicle interface with a manual transmission and external driver
 MinimalBus	Bus of VehicleInterfaces.DriverEnvironment: MinimalBus (minimal set of signals for any type of transmissions)

 BusForAutomaticTransmission	Bus of VehicleInterfaces.DriverEnvironment: BusForAutomaticTransmission
 BusForManualTransmission	Bus of VehicleInterfaces.DriverEnvironment: BusForManualTransmission

VehicleInterfaces.DriverEnvironments.Interfaces.Base

Basic interface definition for the driver-vehicle interface

Information

This partial model defines the interfaces required for the driver environment subsystem within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Connectors

Name	Description
controlBus	Control signal bus
chassisFrame	Chassis reference frame (optional)
steeringWheel	Steering wheel connection (optional)
acceleratorPedal	Accelerator pedal connection (optional)
brakePedal	Brake pedal connection (optional)

VehicleInterfaces.DriverEnvironments.Interfaces.BaseAutomaticTransmission

Interface definition for the driver-vehicle interface with an automatic transmission

Information

Base class for driver models for vehicles with automatic transmissions where the driver control logic is contained within the derived model.

Extends from [Base](#) (Basic interface definition for the driver-vehicle interface).

Connectors

Name	Description
controlBus	Control signal bus
chassisFrame	Chassis reference frame (optional)
steeringWheel	Steering wheel connection (optional)
acceleratorPedal	Accelerator pedal connection (optional)
brakePedal	Brake pedal connection (optional)

VehicleInterfaces.DriverEnvironments.Interfaces.BaseAutomaticTransmissionExternalDriver

Interface definition for the driver-vehicle interface with an automatic transmission and external driver

Information

Base class for driver models for vehicles with automatic transmissions where the driver control logic is

contained in an external driver model based on **VehicleInterfaces.Drivers.Interface.Base**.

Extends from [BaseAutomaticTransmission](#) (Interface definition for the driver-vehicle interface with an automatic transmission).

Parameters

Name	Description
driverInterface	Driver Interaction Bus

Connectors

Name	Description
controlBus	Control signal bus
chassisFrame	Chassis reference frame (optional)
steeringWheel	Steering wheel connection (optional)
acceleratorPedal	Accelerator pedal connection (optional)
brakePedal	Brake pedal connection (optional)
driverInterface	Driver Interaction Bus

VehicleInterfaces.DriverEnvironments.Interfaces.BaseManualTransmission

Interface definition for the driver-vehicle interface with a manual transmission



Information

Base class for driver models for vehicles with manual transmissions where the driver control logic is contained within the derived model.

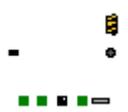
Extends from [Base](#) (Basic interface definition for the driver-vehicle interface).

Connectors

Name	Description
controlBus	Control signal bus
chassisFrame	Chassis reference frame (optional)
steeringWheel	Steering wheel connection (optional)
acceleratorPedal	Accelerator pedal connection (optional)
brakePedal	Brake pedal connection (optional)
clutchPedal	Clutch pedal connection (optional)
shiftConnector	Manual transmission shift connection (optional)

VehicleInterfaces.DriverEnvironments.Interfaces.BaseManualTransmissionExternalDriver

Interface definition for the driver-vehicle interface with a manual transmission and external driver



Information

Base class for driver models for vehicles with manual transmissions where the driver control logic is contained in an external driver model based on **VehicleInterfaces.Drivers.Interface.Base**.

Extends from [BaseManualTransmission](#) (Interface definition for the driver-vehicle interface with a manual transmission).

Connectors

Name	Description
controlBus	Control signal bus
chassisFrame	Chassis reference frame (optional)
steeringWheel	Steering wheel connection (optional)
acceleratorPedal	Accelerator pedal connection (optional)
brakePedal	Brake pedal connection (optional)
clutchPedal	Clutch pedal connection (optional)
shiftConnector	Manual transmission shift connection (optional)
driverInterface	Driver Interaction Bus

[VehicleInterfaces.DriverEnvironments.Interfaces.MinimalBus](#)

Bus of [VehicleInterfaces.DriverEnvironment](#): **MinimalBus** (minimal set of signals for any type of transmissions)



Information

An expandable connector that defines the minimum set of signals required on the **driverBus**.

Extends from [VehicleInterfaces.Interfaces.DriverBus](#) (Bus of [VehicleInterfaces.Interfaces](#): Empty expandable connector used as driver bus), [.VehicleInterfaces.Icons.SignalSubBusWithExplicitSignals](#) (Icon for signal sub-bus where the explicit signals are defined in the bus).

Parameters

Name	Description
acceleratorPedalPosition	Normalized accelerator pedal position (0=fully released ... 1=fully pressed) [1]
brakePedalPosition	Brake pedal position (0=fully released ... 1=fully pressed) [1]
ignition	Engine ignition (Off, On or Start)

Contents

Name	Description
acceleratorPedalPosition	Normalized accelerator pedal position (0=fully released ... 1=fully pressed) [1]
brakePedalPosition	Brake pedal position (0=fully released ... 1=fully pressed) [1]
ignition	Engine ignition (Off, On or Start)

[VehicleInterfaces.DriverEnvironments.Interfaces.BusForAutomaticTransmission](#)

Bus of [VehicleInterfaces.DriverEnvironment](#): **BusForAutomaticTransmission**



Information

An expandable connector that defines the minimum set of signals required on the **driverBus**.

Extends from [VehicleInterfaces.DriverEnvironments.Interfaces.MinimalBus](#) (Bus of [VehicleInterfaces.DriverEnvironment](#): **MinimalBus** (minimal set of signals for any type of transmissions)).

Parameters

Name	Description
acceleratorPedalPosition	Normalized accelerator pedal position (0=fully released ... 1=fully pressed) [1]
brakePedalPosition	Brake pedal position (0=fully released ... 1=fully pressed) [1]
ignition	Engine ignition (Off, On or Start)
requestedGear	Requested gear for automatic transmission if gearboxMode=Manual or =Limited
gearboxMode	Selected gearbox mode (Park, Drive, Neutral, Rear, Manual, Limited)

Contents

Name	Description
acceleratorPedalPosition	Normalized accelerator pedal position (0=fully released ... 1=fully pressed) [1]
brakePedalPosition	Brake pedal position (0=fully released ... 1=fully pressed) [1]
ignition	Engine ignition (Off, On or Start)
requestedGear	Requested gear for automatic transmission if gearboxMode=Manual or =Limited
gearboxMode	Selected gearbox mode (Park, Drive, Neutral, Rear, Manual, Limited)

VehicleInterfaces.DriverEnvironments.Interfaces.BusForManualTransmission

Bus of VehicleInterfaces.DriverEnvironment: BusForManualTransmission



Information

An expandable connector that defines the minimum set of signals required on the **driverBus**.

Extends from [VehicleInterfaces.DriverEnvironments.Interfaces.MinimalBus](#) (Bus of VehicleInterfaces.DriverEnvironment: MinimalBus (minimal set of signals for any type of transmissions)).

Parameters

Name	Description
acceleratorPedalPosition	Normalized accelerator pedal position (0=fully released ... 1=fully pressed) [1]
brakePedalPosition	Brake pedal position (0=fully released ... 1=fully pressed) [1]
ignition	Engine ignition (Off, On or Start)
clutchPedalPosition	Clutch pedal position (0=fully released ... 1=fully pressed) [1]
gear	Selected gear for manual transmission (>0:forward gear, <0:rear gear)

Contents

Name	Description
acceleratorPedalPosition	Normalized accelerator pedal position (0=fully released ... 1=fully pressed) [1]
brakePedalPosition	Brake pedal position (0=fully released ... 1=fully pressed) [1]
ignition	Engine ignition (Off, On or Start)
clutchPedalPosition	Clutch pedal position (0=fully released ... 1=fully pressed) [1]
gear	Selected gear for manual transmission (>0:forward gear, <0:rear gear)

VehicleInterfaces.DriverEnvironments.NoDriverEnvironment

Empty driver-vehicle interface



Information

Empty driver environment. Using this empty model in overall vehicle architecture the functionality of driver environment can be eliminated.

Extends from [VehicleInterfaces.Icons.DriverEnvironment](#) (Icon for a driver environment subsystem), [VehicleInterfaces.Icons.Empty](#) (Icon for an empty component), [Interfaces.Base](#) (Basic interface definition for the driver-vehicle interface).

Connectors

Name	Description
controlBus	Control signal bus
chassisFrame	Chassis reference frame (optional)
steeringWheel	Steering wheel connection (optional)
acceleratorPedal	Accelerator pedal connection (optional)
brakePedal	Brake pedal connection (optional)

VehicleInterfaces.DriverEnvironments.DriveByWireAutomatic**Minimal Drive-by-wire Driver-Vehicle Interface****Information**

Constant acceleration driver with the capability to step between two constant throttle values and to step on/off the brakes.

Extends from [.VehicleInterfaces.Icons.DriverEnvironment](#) (Icon for a driver environment subsystem), [.VehicleInterfaces.DriverEnvironments.Interfaces.BaseAutomaticTransmission](#) (Interface definition for the driver-vehicle interface with an automatic transmission).

Parameters

Name	Description
initialAccelRequest	Initial accelerator pedal position
initialBrakeRequest	Initial brake pedal position
finalAccelRequest	Final accelerator pedal position
finalBrakeRequest	Final brake pedal position
accelTime	Time of accel apply [s]
brakeTime	Time of brake apply [s]
selectedGearboxMode	Current gearbox mode
manualGearRequest	Requested gear in manual mode

Connectors

Name	Description
controlBus	Control signal bus
chassisFrame	Chassis reference frame (optional)
steeringWheel	Steering wheel connection (optional)
acceleratorPedal	Accelerator pedal connection (optional)
brakePedal	Brake pedal connection (optional)

VehicleInterfaces.DriverEnvironments.ConventionalManual

Minimal Driver-Vehicle Interface, manual transmission, uses physical pedal connections



Information

Constant acceleration driver with the capability to step between two constant throttle values. Fixed gear and constant clutch position.

Extends from [VehicleInterfaces.Icons.DriverEnvironment](#) (Icon for a driver environment subsystem), [Interfaces.BaseManualTransmission](#) (Interface definition for the driver-vehicle interface with a manual transmission).

Parameters

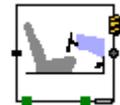
Name	Description
initialAccelRequest	Initial accelerator pedal position
initialBrakeRequest	Initial brake pedal position
finalAccelRequest	Final accelerator pedal position
finalBrakeRequest	Final brake pedal position
accelTime	Time of accel apply [s]
brakeTime	Time of brake apply [s]
selectedGear	Requested gear in manual mode
clutchRequest	Clutch position (constant during simulation)

Connectors

Name	Description
controlBus	Control signal bus
chassisFrame	Chassis reference frame (optional)
steeringWheel	Steering wheel connection (optional)
acceleratorPedal	Accelerator pedal connection (optional)
brakePedal	Brake pedal connection (optional)
clutchPedal	Clutch pedal connection (optional)
shiftConnector	Manual transmission shift connection (optional)

VehicleInterfaces.DriverEnvironments.DriveByWireAutomaticExternalDriver

Minimal drive-by-wire driver-vehicle interface with external driver model



Information

Driver environment that requires an external driver model. Signals received from external driver model are simply propagated through without any modification.

Extends from [VehicleInterfaces.Icons.DriverEnvironment](#) (Icon for a driver environment subsystem), [Interfaces.BaseAutomaticTransmissionExternalDriver](#) (Interface definition for the driver-vehicle interface with an automatic transmission and external driver).

Parameters

Name	Description
------	-------------

driverInterface	Driver Interaction Bus
-----------------	------------------------

Connectors

Name	Description
controlBus	Control signal bus
chassisFrame	Chassis reference frame (optional)
steeringWheel	Steering wheel connection (optional)
acceleratorPedal	Accelerator pedal connection (optional)
brakePedal	Brake pedal connection (optional)
driverInterface	Driver Interaction Bus

VehicleInterfaces.Drivers

Collection of driver subsystem definitions

Information

The driver subsystem interfaces are defined in this sub-package of the VehicleInterfaces library. The driver subsystem has the following connector:

- **driverInterface** - a DriverInterface connector which is an empty expandable connector

Effects to be modelled in this subsystem

The driver logic should be modelled in this subsystem and communication with the DriverEnvironment subsystem should be via normalised control signals.

Extends from [VehicleInterfaces.Icons.VariantLibrary](#) (Icon for a package class which contains several variants of one assembly or component).

Package Content

Name	Description
 Tutorial	Drivers Tutorial
 Interfaces	Collection of interface definitions for driver
 NoDriver	Empty driver
 MinimalDriver	Constant acceleration Driver

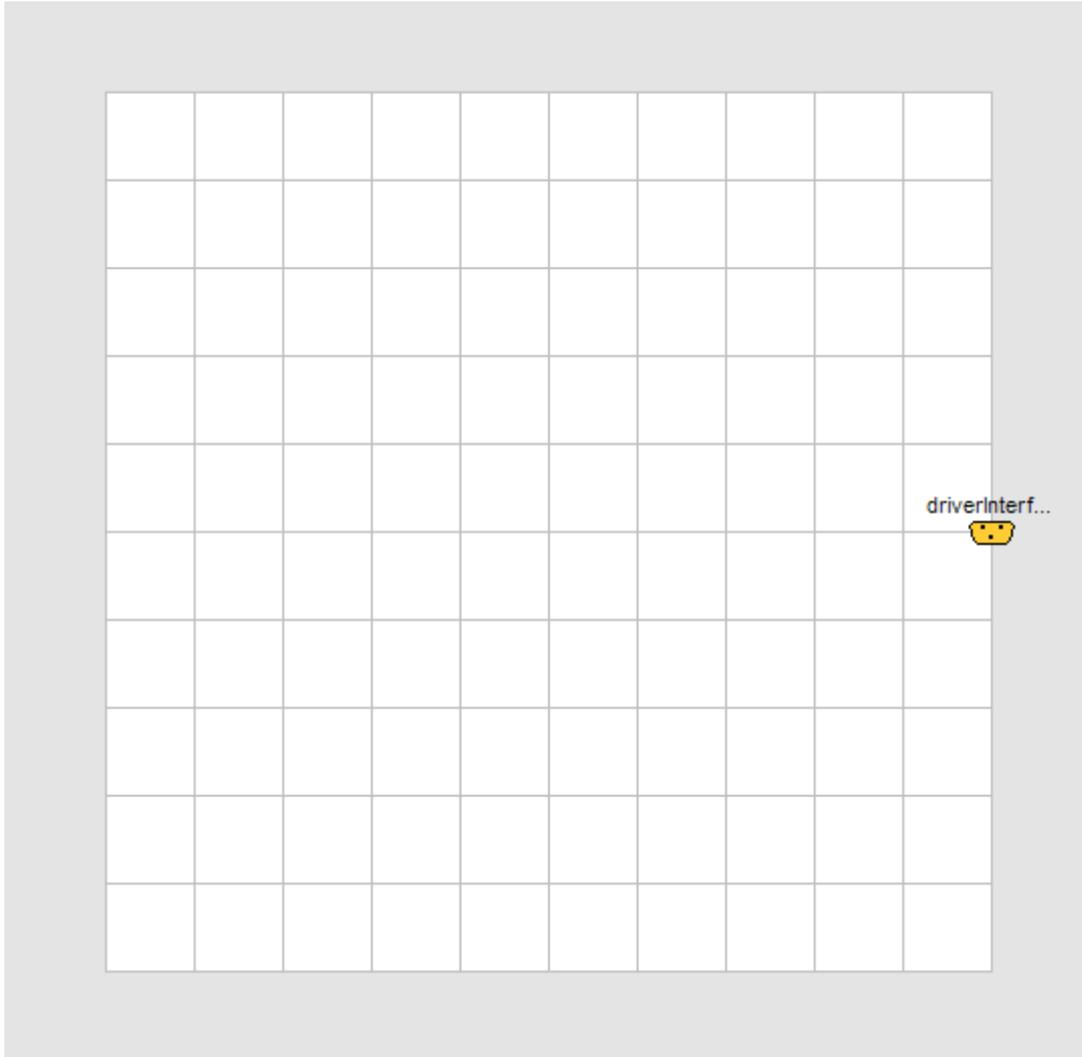
VehicleInterfaces.Drivers.Tutorial

Tutorial - Defining a new driver model

The following process will demonstrate how to create a new driver model using these interface definitions. This tutorial will guide you through building a driver for a conventional automatic transmission passenger car.



1. Create a new model that extends **VehicleInterfaces.Drivers.Interfaces.Base**, it should look like this:



2. Unlike the other interface definitions in the VehicleInterfaces package the Driver model doesn't include optional connections. It simply uses an expandable connector to exchange information with the DriverEnvironment subsystem. The required signals are defined in the [Driver interaction bus](#) section of the Users Guide. Normalised mechanical sensors and actuators are provided in the [VehicleInterfaces.Mechanics](#) package.

VehicleInterfaces.Drivers.Interfaces

Collection of interface definitions for driver

Information

A collection of partial base classes which define interfaces for driver models.

Extends from Modelica.Icons.InterfacesPackage (Icon for packages containing interfaces).

Package Content

Name	Description
. Base	Basic interface definition for a driver
 MinimalBus	Bus of VehicleInterfaces.Drivers: MinimalBus (minimal set of signals for

	any type of transmissions)
 BusForAutomaticTransmission	Bus of VehicleInterfaces.Drivers: BusForAutomaticTransmission
 BusForManualTransmission	Bus of VehicleInterfaces.Drivers: BusForManualTransmission

VehicleInterfaces.Drivers.Interfaces.Base

Basic interface definition for a driver

Information

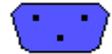
This partial model defines the interfaces required for the driver subsystem within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Connectors

Name	Description
driverInterface	Driver Interaction Bus

VehicleInterfaces.Drivers.Interfaces.MinimalBus

Bus of VehicleInterfaces.Drivers: MinimalBus (minimal set of signals for any type of transmissions)



Information

An expandable connector that defines the minimum set of signals required on the **driverInteractionBus** for a driver of an automatic vehicle.

Extends from [.VehicleInterfaces.Interfaces.DriverInterface](#) (Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as driver interface bus), [.VehicleInterfaces.Icons.SignalSubBusWithExplicitSignals](#) (Icon for signal sub-bus where the explicit signals are defined in the bus).

Parameters

Name	Description
acceleratorPedal	Accelerator pedal
brakePedal	Brake pedal
steeringWheel	Steering wheel
ignition	Engine ignition (Off, On or Start)
vehicleSpeed	Vehicle speed [m/s]
engineSpeed	Engine speed [rad/s]

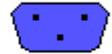
Contents

Name	Description
acceleratorPedal	Accelerator pedal
brakePedal	Brake pedal
steeringWheel	Steering wheel
ignition	Engine ignition (Off, On or Start)
vehicleSpeed	Vehicle speed [m/s]

engineSpeed	Engine speed [rad/s]
-------------	----------------------

VehicleInterfaces.Drivers.Interfaces.BusForAutomaticTransmission

Bus of VehicleInterfaces.Drivers: BusForAutomaticTransmission



Information

An expandable connector that defines the minimum set of signals required on the **driverInteractionBus** for a driver of an automatic vehicle.

Extends from [VehicleInterfaces.Drivers.Interfaces.MinimalBus](#) (Bus of VehicleInterfaces.Drivers: MinimalBus (minimal set of signals for any type of transmissions)).

Parameters

Name	Description
acceleratorPedal	Accelerator pedal
brakePedal	Brake pedal
steeringWheel	Steering wheel
ignition	Engine ignition (Off, On or Start)
vehicleSpeed	Vehicle speed [m/s]
engineSpeed	Engine speed [rad/s]
requestedGear	Requested gear for automatic transmission if gearboxMode=Manual or =Limited
gearboxMode	Selected gearbox mode (Park, Drive, Neutral, Rear, Manual, Limited)

Contents

Name	Description
acceleratorPedal	Accelerator pedal
brakePedal	Brake pedal
steeringWheel	Steering wheel
ignition	Engine ignition (Off, On or Start)
vehicleSpeed	Vehicle speed [m/s]
engineSpeed	Engine speed [rad/s]
requestedGear	Requested gear for automatic transmission if gearboxMode=Manual or =Limited
gearboxMode	Selected gearbox mode (Park, Drive, Neutral, Rear, Manual, Limited)

VehicleInterfaces.Drivers.Interfaces.BusForManualTransmission

Bus of VehicleInterfaces.Drivers: BusForManualTransmission



Information

An expandable connector that defines the minimum set of signals required on the **driverInteractionBus** for a driver of a manual vehicle.

Extends from [VehicleInterfaces.Drivers.Interfaces.MinimalBus](#) (Bus of VehicleInterfaces.Drivers: MinimalBus (minimal set of signals for any type of transmissions)).

Parameters

Name	Description
acceleratorPedal	Accelerator pedal
brakePedal	Brake pedal
steeringWheel	Steering wheel
ignition	Engine ignition (Off, On or Start)
vehicleSpeed	Vehicle speed [m/s]
engineSpeed	Engine speed [rad/s]
clutchPedal	Clutch pedal
gear	Selected gear for manual transmission (>0:forward gear, <0:rear gear)
clutchLocked	Clutch locked flag

Contents

Name	Description
acceleratorPedal	Accelerator pedal
brakePedal	Brake pedal
steeringWheel	Steering wheel
ignition	Engine ignition (Off, On or Start)
vehicleSpeed	Vehicle speed [m/s]
engineSpeed	Engine speed [rad/s]
clutchPedal	Clutch pedal
gear	Selected gear for manual transmission (>0:forward gear, <0:rear gear)
clutchLocked	Clutch locked flag

VehicleInterfaces.Drivers.NoDriver**Empty driver****Information**

Empty driver model. Using this empty model in overall vehicle architecture the functionality of driver can be eliminated.

Extends from [VehicleInterfaces.Drivers.Interfaces.Base](#) (Basic interface definition for a driver), [VehicleInterfaces.Icons.Driver](#) (Icon for a driver subsystem), [VehicleInterfaces.Icons.Empty](#) (Icon for an empty component).

Connectors

Name	Description
driverInterface	Driver Interaction Bus

VehicleInterfaces.Drivers.MinimalDriver**Constant acceleration Driver**

Information

Constant acceleration driver with the capability to step between two constant throttle values.

Extends from [VehicleInterfaces.Icons.Driver](#) (Icon for a driver subsystem),
[VehicleInterfaces.Drivers.Interfaces.Base](#) (Basic interface definition for a driver).

Parameters

Name	Description
initialAccelRequest	Initial accelerator pedal position
initialBrakeRequest	Initial brake pedal position
finalAccelRequest	Final accelerator pedal position
finalBrakeRequest	Final brake pedal position
accelTime	Time of accel apply [s]
brakeTime	Time of brake apply [s]
selectedGearboxMode	Current gearbox mode
manualGearRequest	Requested gear in manual mode

Connectors

Name	Description
driverInterface	Driver Interaction Bus

VehicleInterfaces.ElectricDrives

Collection of electric drive subsystem definitions

Information

The electric drive subsystem interfaces are defined in this sub-package of the VehicleInterfaces library. The electric drive subsystem has the following connectors:

- **shaft** - 1D rotational connection to the driven system
- **controlBus** - control signal bus connection

Extends from [VehicleInterfaces.Icons.VariantLibrary](#) (Icon for a package class which contains several variants of one assembly or component).

Package Content

Name	Description
 Tutorial	Electric Drives Tutorial
 Interfaces	Collection of interface definitions for electric drive
 SimpleMotorDC	Simple DC electric motor

VehicleInterfaces.ElectricDrives.Tutorial

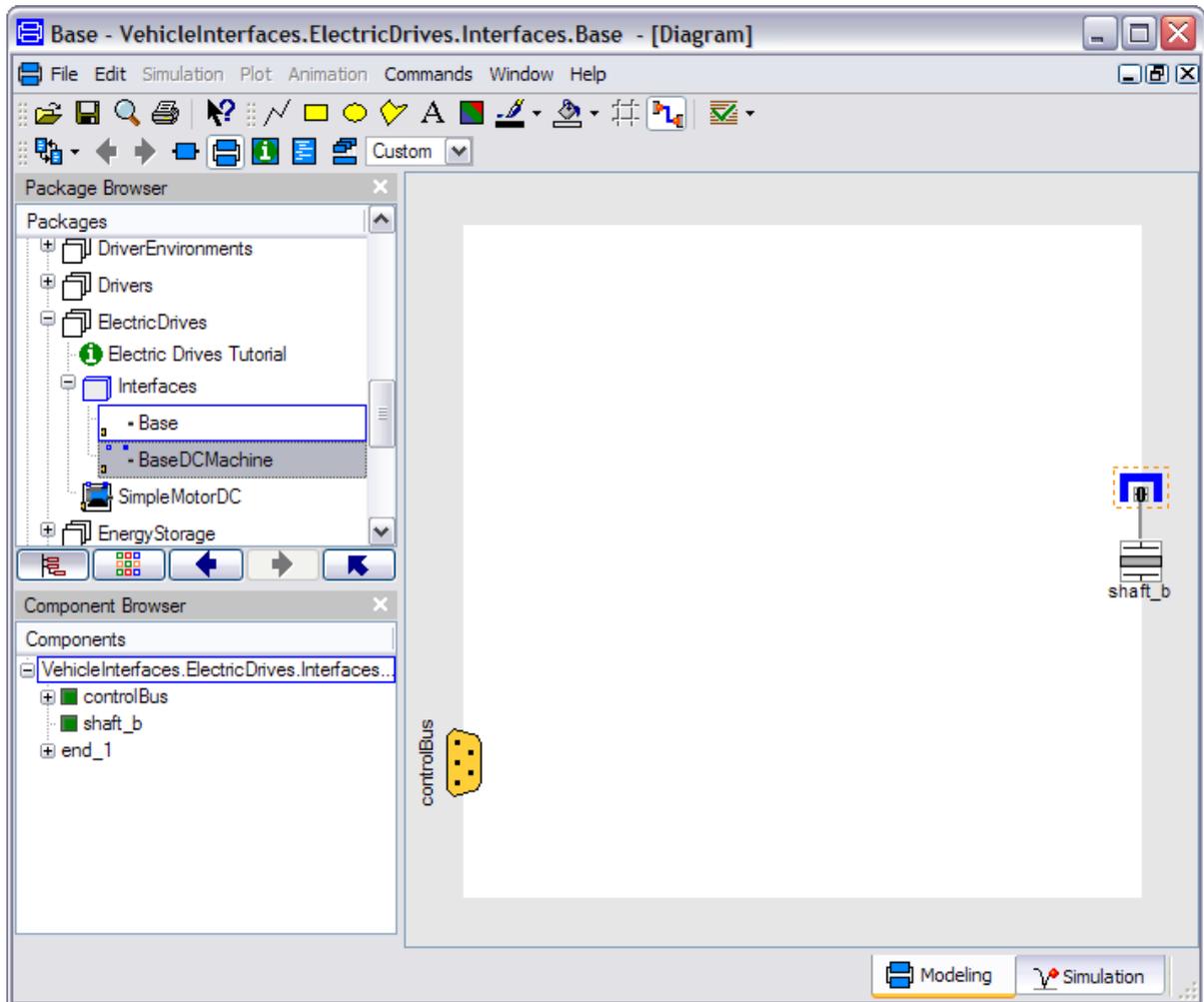
Tutorial - Defining a new electric motor model

The following process will demonstrate how to create a new electric motor model using these interface definitions. This tutorial will guide you through building a simple DC electric motor.

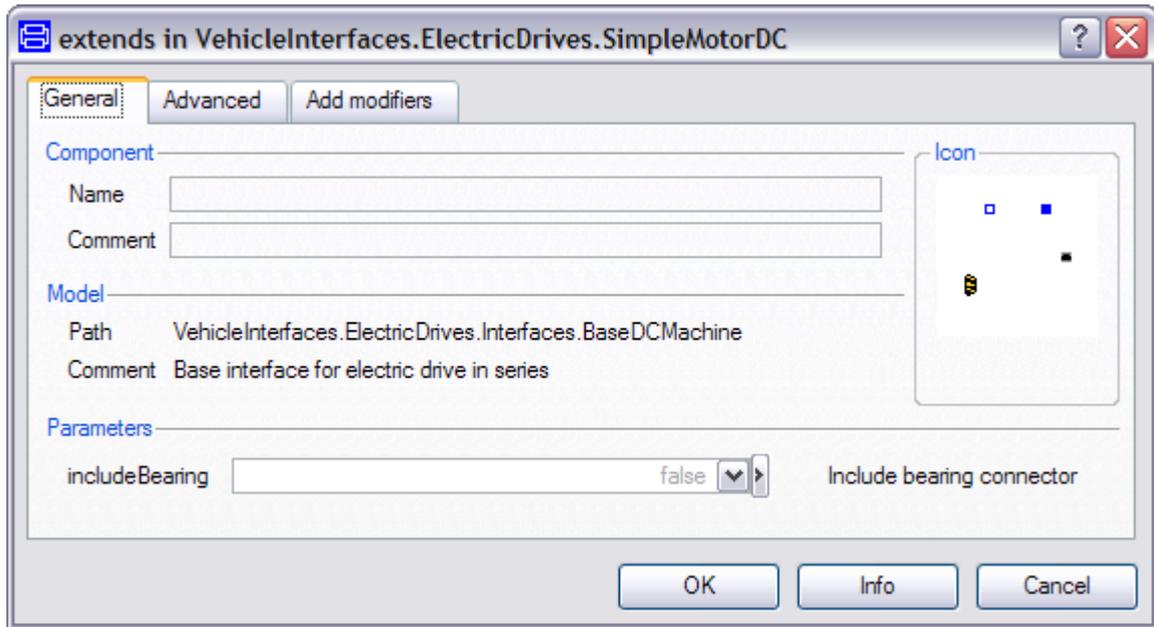
1. Create a new model that extends **VehicleInterfaces.ElectricDrives.Interfaces.BaseDCMachine**, it



should look like this:



2. In the component browser, right click on **BaseDCMachine** and select **Parameters** from the context menu to produce the following parameter dialog



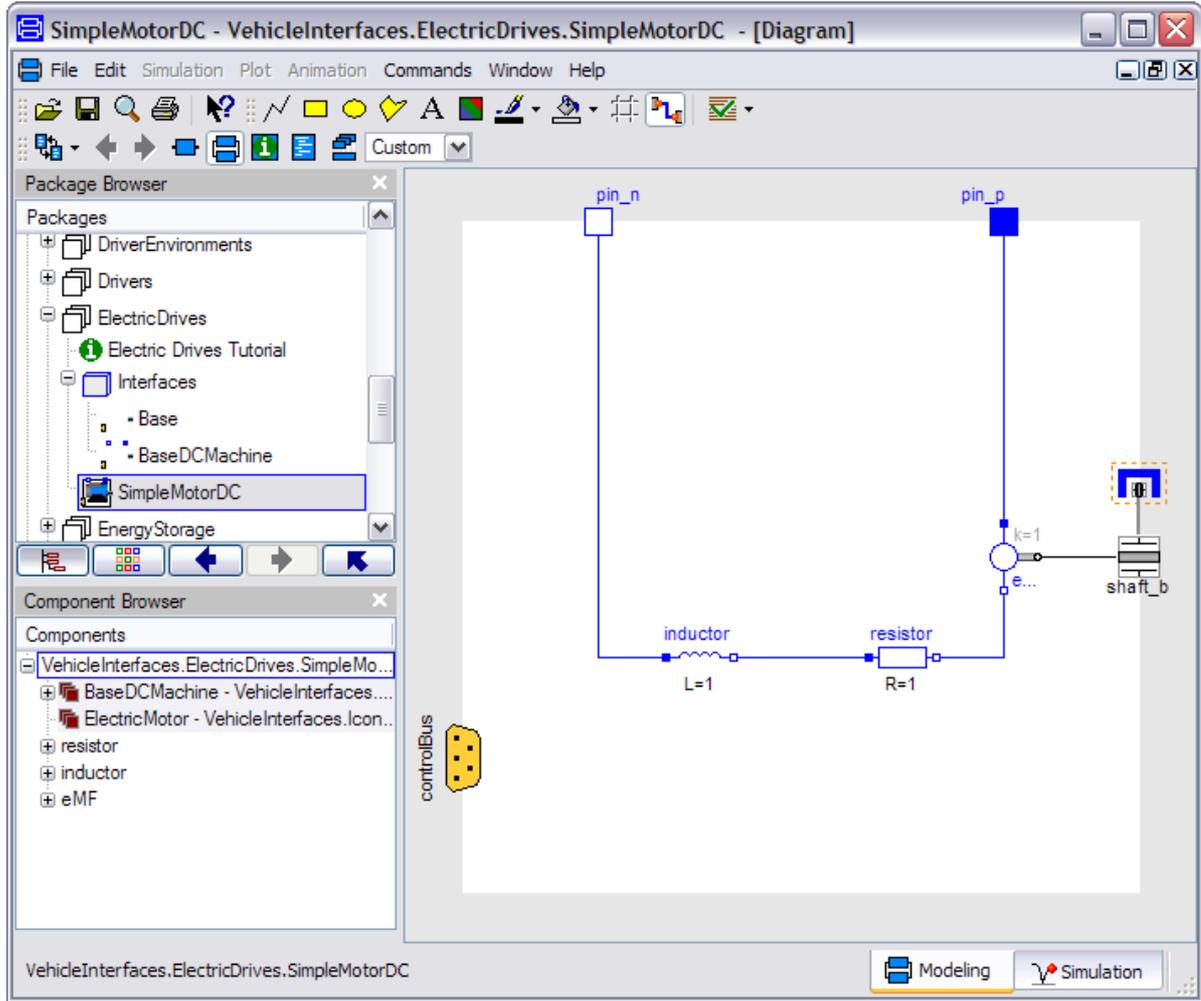
3. This dialog allows you to enable/disable the optional connections by setting **includeBearing** as required for your new motor model. The shaft_b connector is of the type [VehicleInterfaces.Interfaces.FlangeWithBearing](#), the parameter **includeBearing** controls whether the bearing connector within the shaft_b connector is enabled or not.
4. You can now define your driveline model as required

Creating a simple electric motor example

The following steps demonstrate how to create a simple rear-wheel drive driveline model. The driveline model will transmit the torque from the transmission to the rear wheels via a propshaft, differential with final drive and then two halfshafts. No torque reaction in to the transmission housings will be modelled.

Starting from step 3 above.

1. First, decide if the bearing connector is required in the model and set the internal parameter appropriately.
2. Add the following blocks and connections to the diagram.



VehicleInterfaces.ElectricDrives.Interfaces

Collection of interface definitions for electric drive

Information

A collection of partial base classes which define interfaces for electric drive models. Extends from Modelica.Icons.InterfacesPackage (Icon for packages containing interfaces).

Package Content

Name	Description
- Base	Basic interface for an electric drives
- BaseDCMachine	Base interface for electric drive in series

VehicleInterfaces.ElectricDrives.Interfaces.Base

Basic interface for an electric drives

Information

This partial model defines the interfaces required for an electric machine subsystem within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Parameters

Name	Description
Advanced	
usingMultiBodySystem	=true if connecting to a MultiBody system with a 1D model

Connectors

Name	Description
controlBus	Control signal bus
shaft_b	Mechanical shaft

VehicleInterfaces.ElectricDrives.Interfaces.BaseDCMachine

Base interface for electric drive in series



Information

This partial model defines the interfaces required for an electric machine subsystem within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Extends from [VehicleInterfaces.ElectricDrives.Interfaces.Base](#) (Basic interface for an electric drives).

Parameters

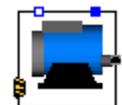
Name	Description
Advanced	
usingMultiBodySystem	=true if connecting to a MultiBody system with a 1D model

Connectors

Name	Description
controlBus	Control signal bus
shaft_b	Mechanical shaft
pin_p	Positive electrical pin
pin_n	Negative electrical pin

VehicleInterfaces.ElectricDrives.SimpleMotorDC

Simple DC electric motor



Information

A simple DC electric motor with inductance and internal resistance.

Extends from [Interfaces.BaseDCMachine](#) (Base interface for electric drive in series), [VehicleInterfaces.Icons.ElectricMotor](#) (Icon for an electric drive subsystem).

Parameters

Name	Description
L	Inductance [H]
R	Resistance [Ohm]
k	Transformation coefficient [N.m/A]
Advanced	
usingMultiBodySystem	=true if connecting to a MultiBody system with a 1D model

Connectors

Name	Description
controlBus	Control signal bus
shaft_b	Mechanical shaft
pin_p	Positive electrical pin
pin_n	Negative electrical pin

VehicleInterfaces.EnergyStorage

Collection of energy storage subsystem definitions

Information

The energy storage subsystem interfaces are defined in this sub-package of the VehicleInterfaces library. The energy storage subsystem has the following connectors:

- **pin_p** - Electrical connection
- **pin_n** - Electrical connection
- **controlBus** - control signal bus connection

Extends from [VehicleInterfaces.Icons.VariantLibrary](#) (Icon for a package class which contains several variants of one assembly or component).

Package Content

Name	Description
 Tutorial	Energy Storage Device Tutorial
 Interfaces	Collection of interface definitions for energy storage
 Battery	Simple battery

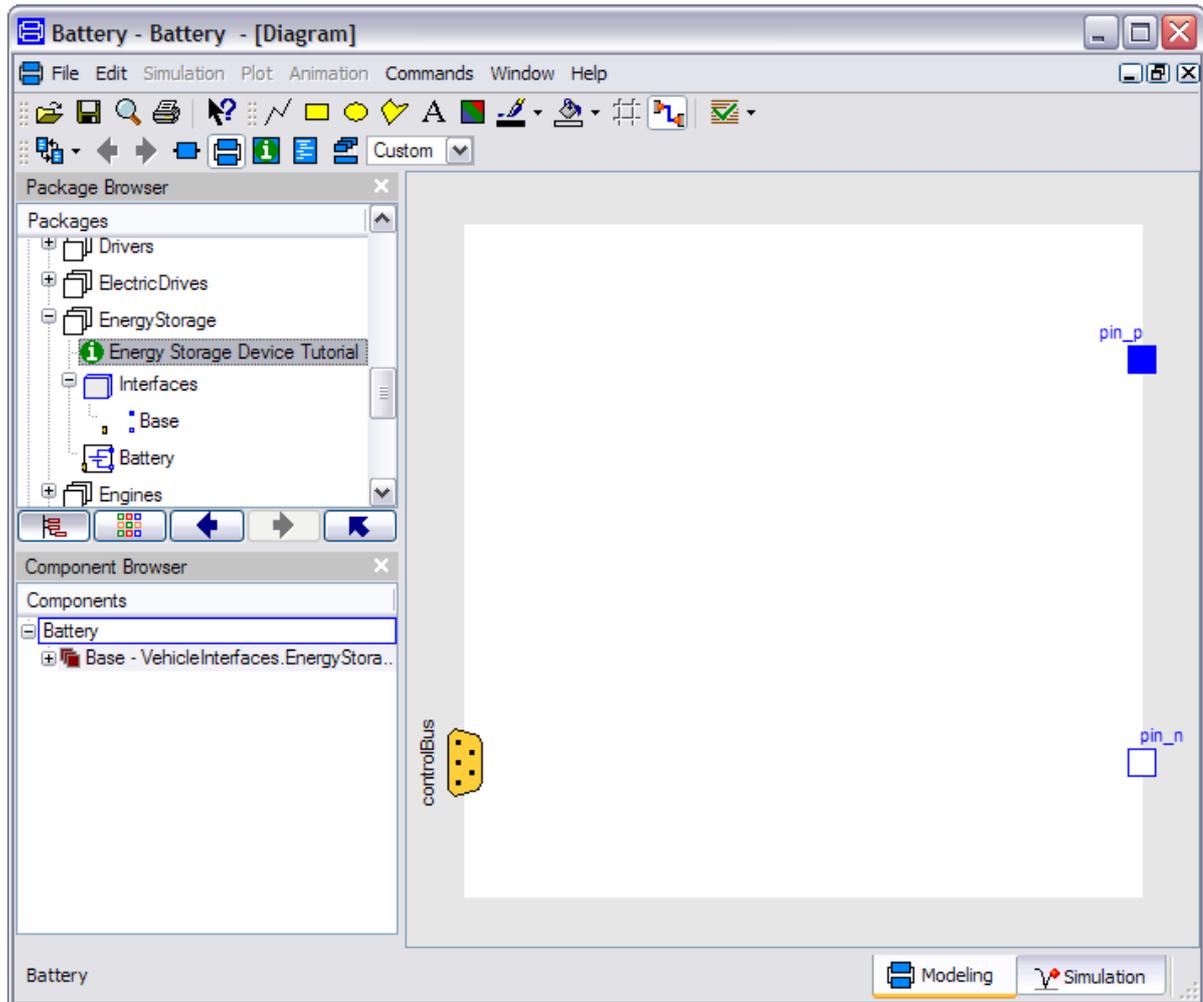
VehicleInterfaces.EnergyStorage.Tutorial

Tutorial - Defining a new energy storage device model

The following process will demonstrate how to create a new energy storage device model using these interface definitions. This tutorial will guide you through building a simple battery.



1. Create a new model that extends **VehicleInterfaces.EnergyStorage.Interfaces.Base**, it should look like this:



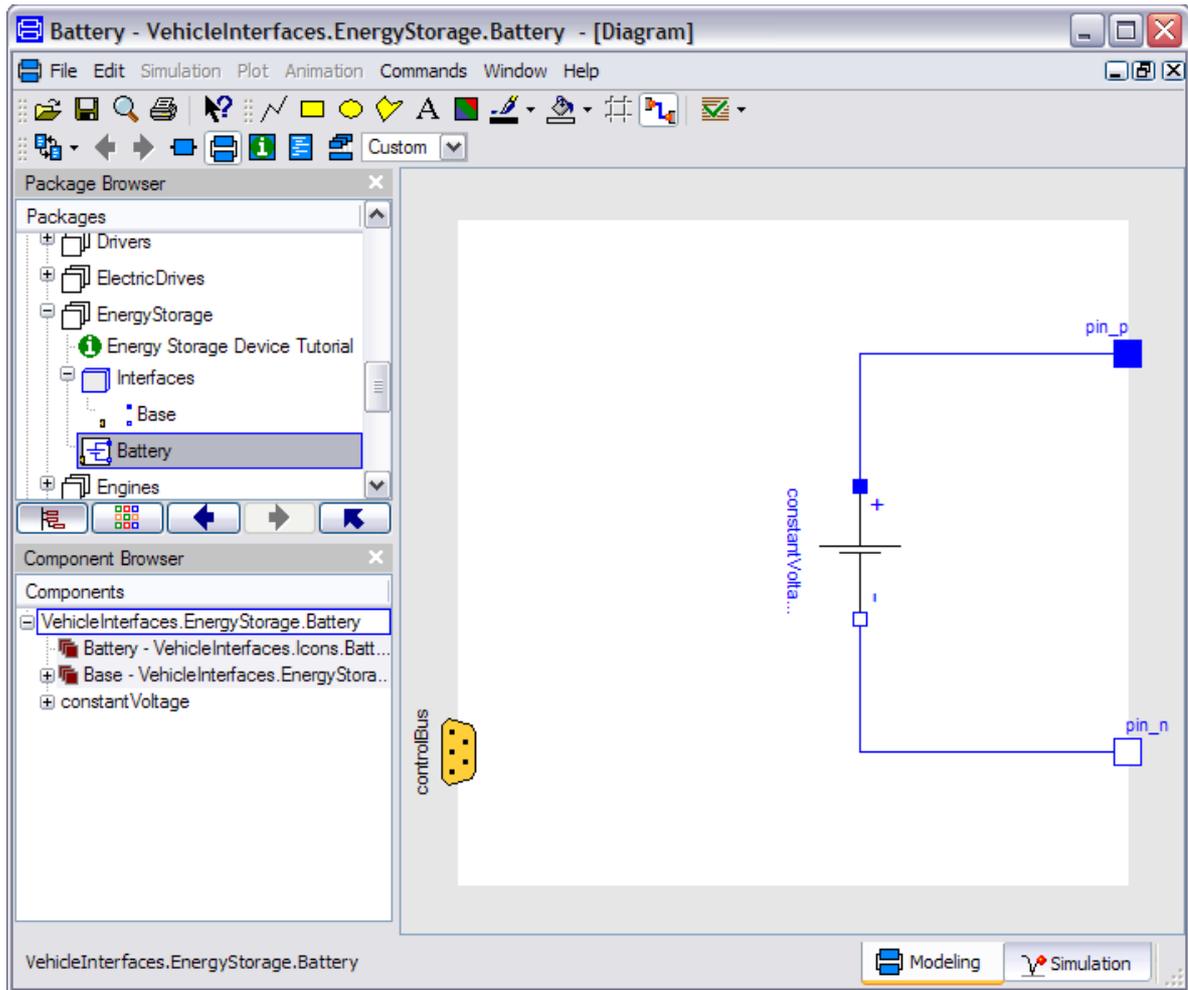
2. Unlike most of the other systems in the VehicleInterfaces package, the energy storage device interface definition does not contain any optional connectors.
3. You can now define your energy storage device model as required

Creating a simple battery example

The following steps demonstrate how to create a simple battery model. The battery model will have a constant voltage and an internal resistance.

Starting from step 2 above.

1. Add the following blocks and connections to the diagram.



VehicleInterfaces.EnergyStorage.Interfaces

Collection of interface definitions for energy storage

Information

A collection of partial base classes which define interfaces for energy storage models. Extends from Modelica.Icons.InterfacesPackage (Icon for packages containing interfaces).

Package Content

Name	Description
Base	Basic interface for an energy storage device

VehicleInterfaces.EnergyStorage.Interfaces.Base

Basic interface for an energy storage device

Information

This partial model defines the interfaces required for an electric machine subsystem within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Parameters

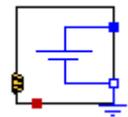
Name	Description
includeGround	Include inner electrical ground at pin_n
includeHeatPort	Include the battery heat port

Connectors

Name	Description
controlBus	Control signal bus
pin_p	Positive electrical pin
pin_n	Negative electrical pin
heatPort	Conditional port for exhaust heat flow

VehicleInterfaces.EnergyStorage.Battery

Simple battery



Information

An ideal battery with a constant voltage. No state of charge is introduced.

Extends from [.VehicleInterfaces.Icons.Battery](#) (Icon for a battery subsystem),
[.VehicleInterfaces.EnergyStorage.Interfaces.Base](#) (Basic interface for an energy storage device).

Parameters

Name	Description
V	Value of constant voltage [V]
includeGround	Include inner electrical ground at pin_n
includeHeatPort	Include the battery heat port

Connectors

Name	Description
controlBus	Control signal bus
pin_p	Positive electrical pin
pin_n	Negative electrical pin

VehicleInterfaces.Engines

Collection of combustion engine subsystem definitions

Information

The engine subsystem interfaces are defined in this sub-package of the VehicleInterfaces library. The engine subsystem has the following connectors some of which are optional (see below for more information):

- **accessoryFlange** - 1D rotational connection to the accessories subsystem (or other components driven by the engine)
- **transmissionFlange** - 1D rotational connection to the transmission subsystem
- **controlBus** - control signal bus connection
- **acceleratorPedal** - 1D translational connection for the accelerator pedal connection to the

- driverEnvironment (optional)
- **engineMount** - MultiBody connection to transmit the engine mount reactions (optional)

The optional connectors are, by default, disabled and can be ignored if not required. They can be enabled by setting the appropriate parameter to be true. This is only possible at design time, i.e. when you are building the subsystem model.

Effects to be modelled in this subsystem

Within the VehicleInterfaces package the engine subsystem is used to model the generation of torque and the application of this torque to the transmission and accessories flange. The connections to the transmission and accessories subsystems are via 1D rotational connectors. The torque reaction in to the engine block and the block itself are also to be modelled in this subsystem if required. The torque reactions, if included, should all be referred back to a single reference frame for the engine block (the engineMount connector).

Extends from [VehicleInterfaces.Icons.VariantLibrary](#) (Icon for a package class which contains several variants of one assembly or component).

Package Content

Name	Description
 Tutorial	Engines Tutorial
 Interfaces	Collection of interface definitions for engine
 NullEngine	Empty engine
 MinimalEngine	Simple engine model with torque proportional to accelerator pedal position
 MinimalEngineUsingPedal	Simple engine model with torque proportional to accelerator pedal position, uses accelerator pedal connection

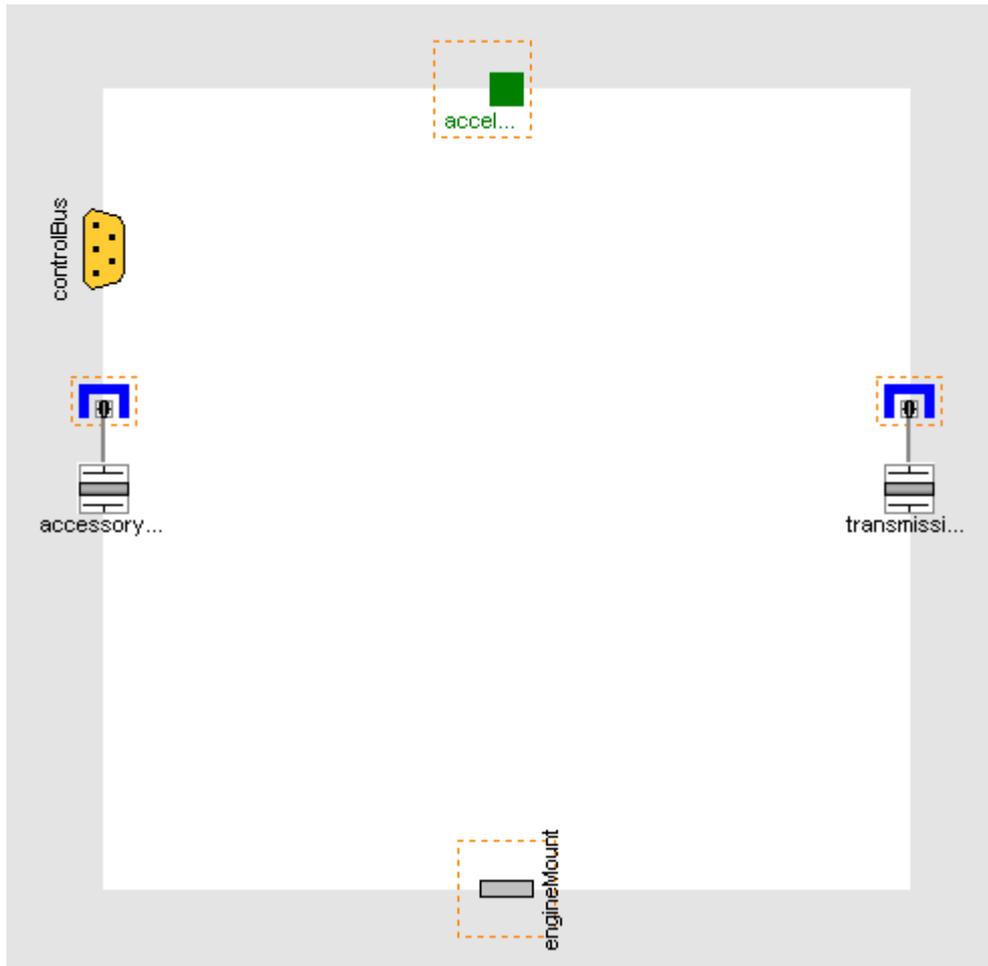
VehicleInterfaces.Engines.Tutorial

Tutorial - Defining a new engine model

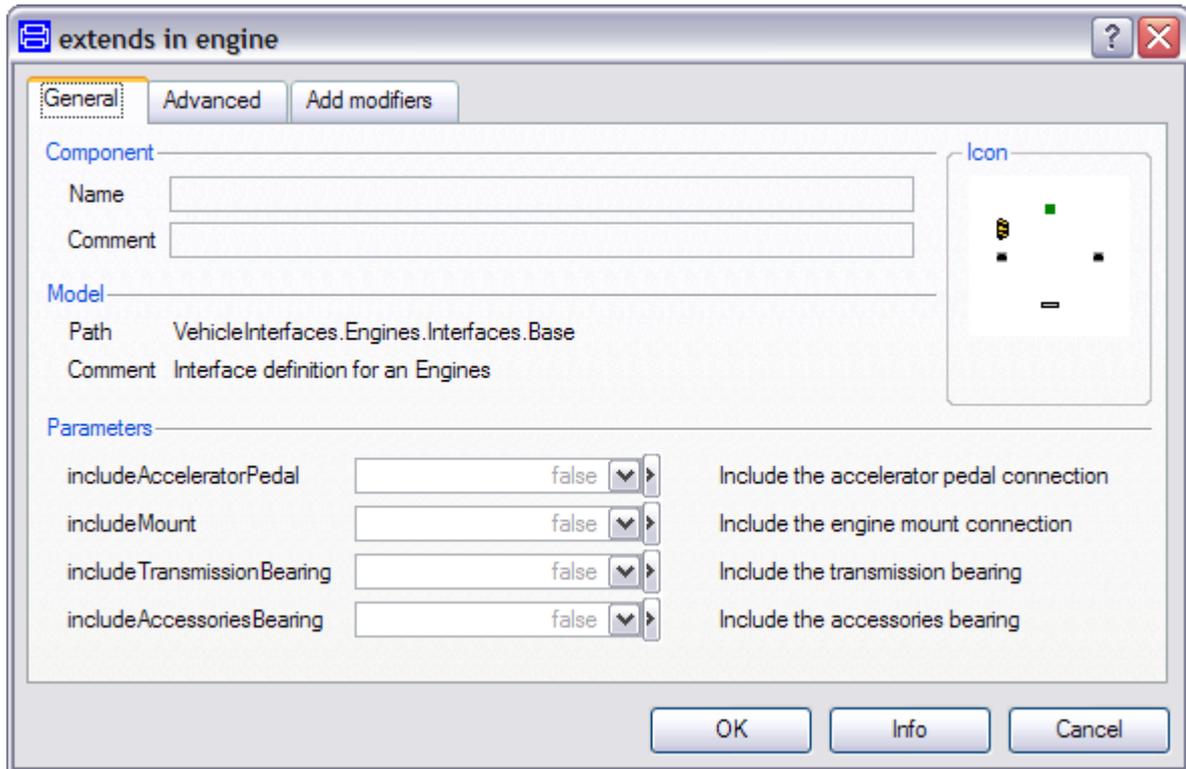
The following process will demonstrate how to create a new engine model using this interface definition.



1. Create a new model that extends **VehicleInterfaces.Engines.Interfaces.Base**, it should look like this:



2. In the component browser, right click on **Base** and select **Parameters** from the context menu to produce the following parameter dialog



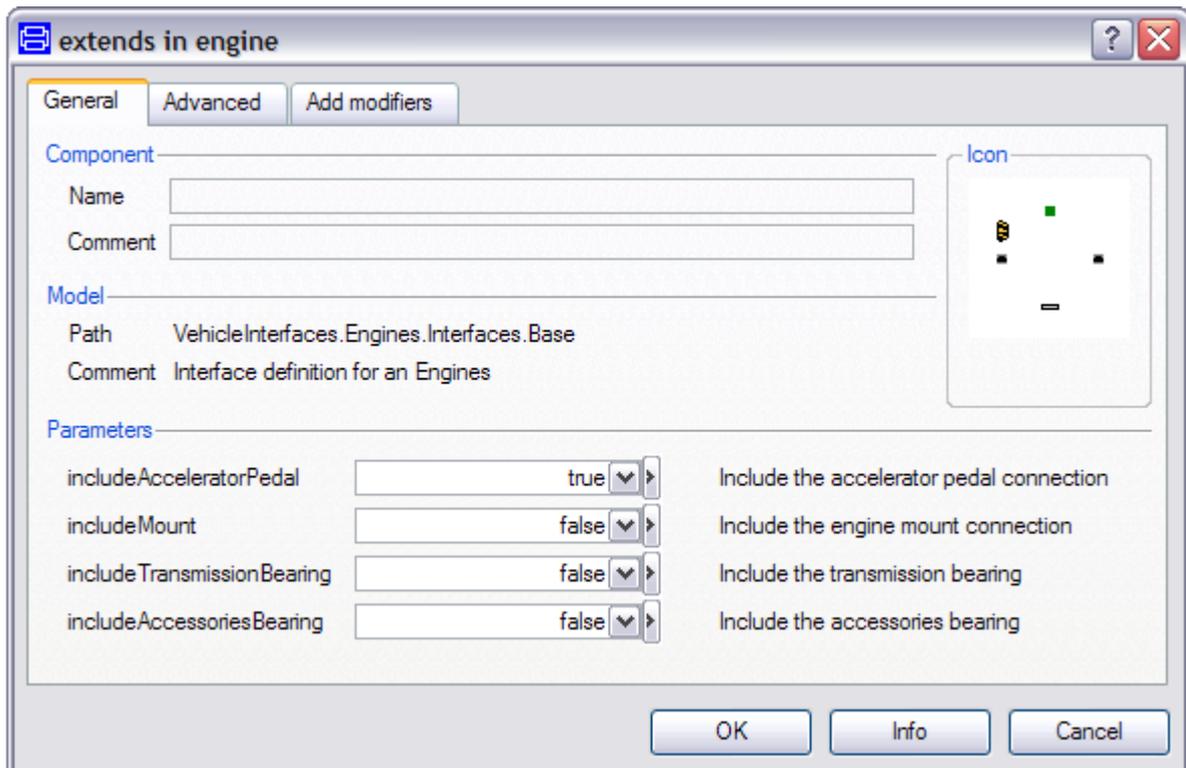
3. This dialog allows you to enable/disable the optional connections by setting **includeAcceleratorPedal** and **includeMount** as required for your new engine model. The **accessoriesFlange** and **transmissionFlange** are of the type [VehicleInterfaces.Interfaces.FlangeWithBearing](#), the parameters **includeTransmissionBearing** and **includeAccessoriesBearing** controls whether the bearing connectors within these connections are enabled or not.
4. You can now define your engine model as required

Creating the MinimalEngine example

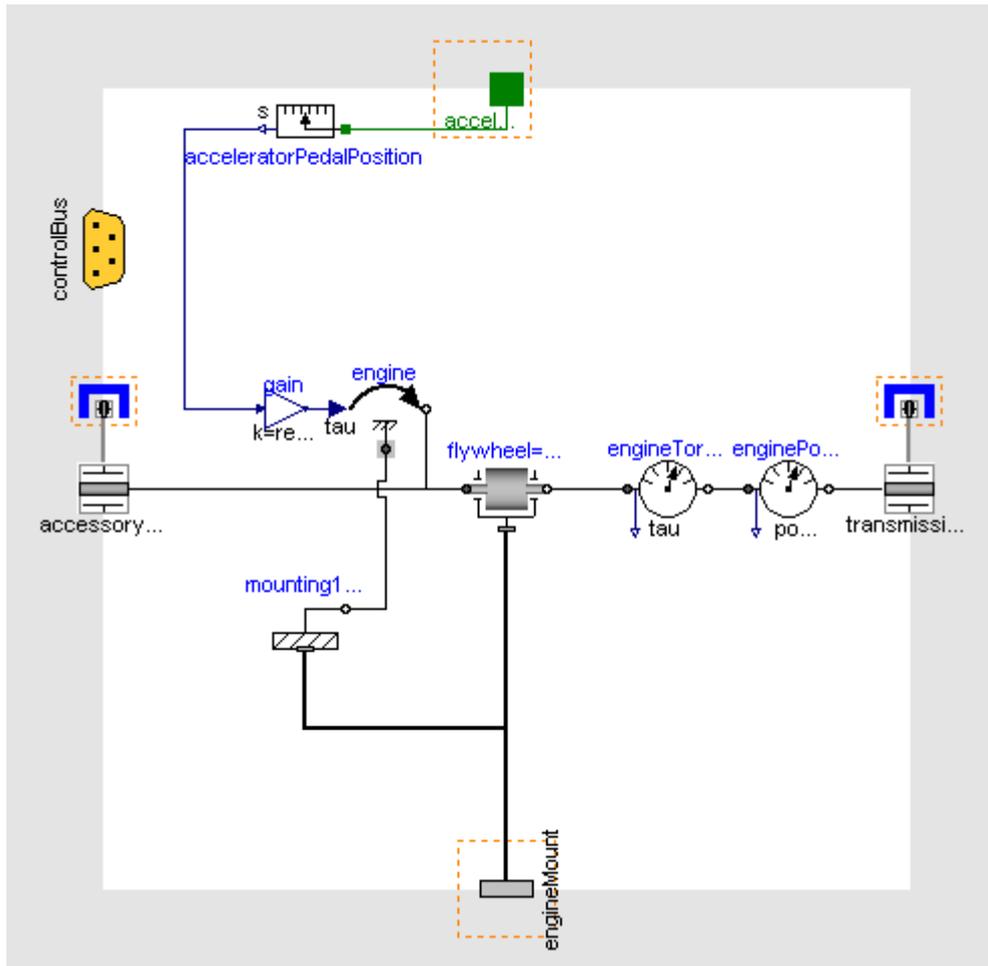
The following steps demonstrate how to create a simple engine model. The engine model will apply torque at the flywheel inertia based on a simple gain from the driver's accelerator pedal. No torque reaction in to the engine block will be modelled.

Starting from step 3 above.

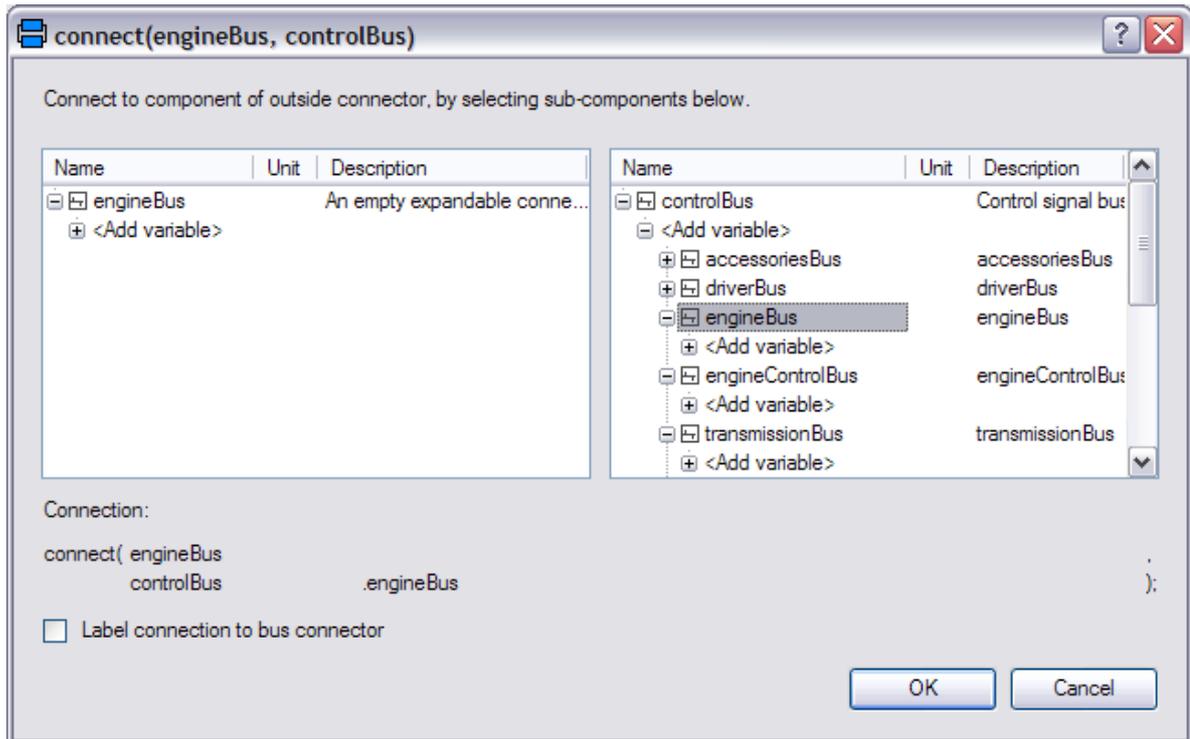
1. First, decide which of the optional connectors are required to model. For this example we need the accelerator pedal connection but not the engine mount connection



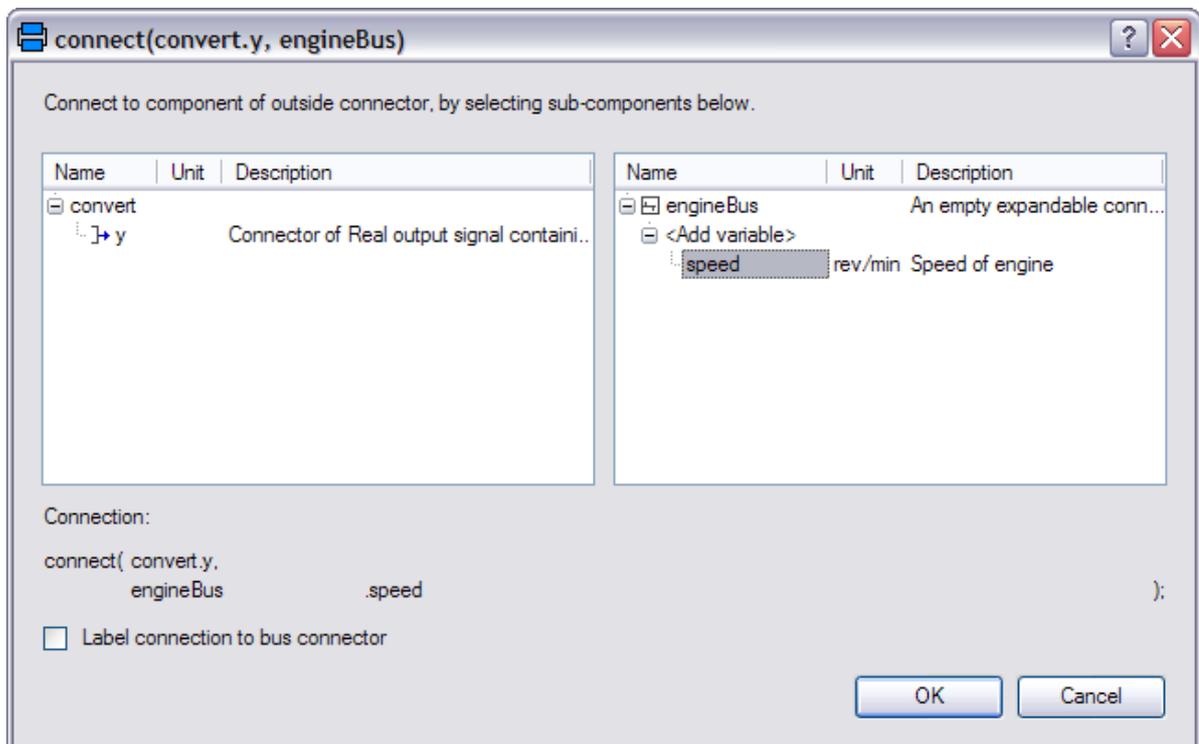
2. Add the following blocks and connections to the diagram



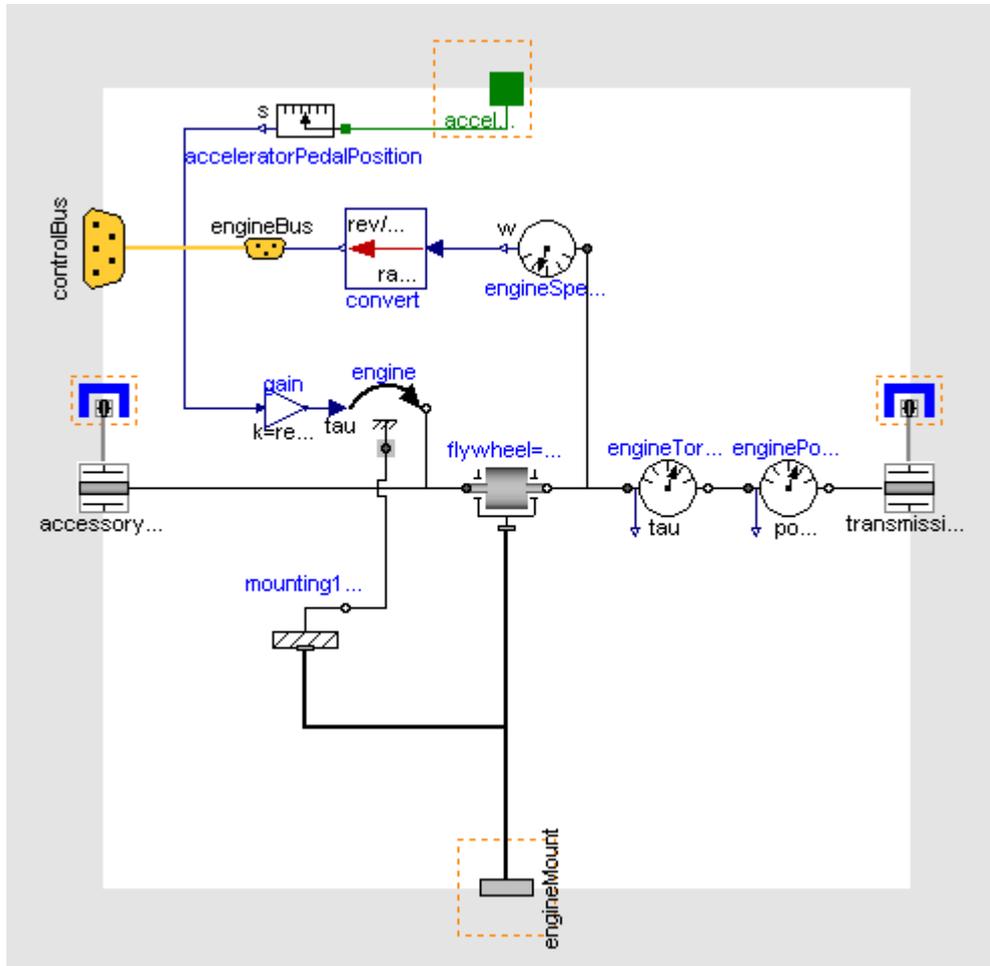
- Next, we need to check to see if any connections to the control signal bus are required for the engine, see [here](#) for a complete list of the minimum connections required. In this case we need to add the engine speed to the control signal bus and this can be done by connecting a speed sensor to the flywheel and then connecting this to the signal bus. As the engine speed signal is added to the engineBus we first need to add this connector. The engineBus connector is **VehicleInterfaces.Interfaces.EngineBus**. This should be connected to the **controlBus**, when this connection is made the following dialog is produced and should be completed as shown.



- We shouldn't connect the speed sensor directly to the engineBus connector though because the units for the speed signal would be incorrect. The definition of the speed signal on the control bus also states that this should be in **rpm** but the speed sensor measures speed in **rad/s**. We can convert the units using the conversion blocks that can be found in **Modelica.Blocks.Math.UnitConversions**. Add a conversion block to convert the output of the speed sensor to rpm and connect this to the engineBus. When this connection is made the following dialog will be produced and should be complete as shown.



- The model is now complete and should check successfully and can be used in any model compatible with the VehicleInterfaces library assuming the selected Driver model also uses the accelerator pedal connection



VehicleInterfaces.Engines.Interfaces

Collection of interface definitions for engine

Information

A collection of partial base classes which define interfaces for engine models.

Extends from Modelica.Icons.InterfacesPackage (Icon for packages containing interfaces).

Package Content

Name	Description
 Base	Basic interface definition for an engine
 StandardBus	Bus of VehicleInterfaces.Engines: StandardBus of signals generated by the Engine

VehicleInterfaces.Engines.Interfaces.Base

Basic interface definition for an engine



Information

This partial model defines the interfaces required for an engine subsystem within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Parameters

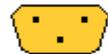
Name	Description
Advanced	
usingMultiBodyAccessories	=true if using MultiBody accessories with a 1D engine
usingMultiBodyTransmission	=true if using a MultiBody transmission with a 1D engine

Connectors

Name	Description
accessoryFlange	Connection for the engine accessories
transmissionFlange	Connection to the transmission
controlBus	Control signal bus
engineMount	Engine mount connection (optional)
acceleratorPedal	Accelerator pedal connection (optional)

VehicleInterfaces.Engines.Interfaces.StandardBus

Bus of VehicleInterfaces.Engines: StandardBus of signals generated by the Engine



Information

Bus with a set of standard signals generated by the engine subsystem.

Extends from [.VehicleInterfaces.Interfaces.EngineBus](#) (Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as engine bus), [.VehicleInterfaces.Icons.SignalSubBusWithExplicitSignals](#) (Icon for signal sub-bus where the explicit signals are defined in the bus).

Parameters

Name	Description
speed	Speed of engine [rad/s]

Contents

Name	Description
speed	Speed of engine [rad/s]

VehicleInterfaces.Engines.NullEngine

Empty engine



Information

Empty engine model (just rigid connection between accessory and transmission flange). Using this empty model in overall vehicle architecture the functionality of engine can be eliminated.

Extends from [VehicleInterfaces.Icons.Engine](#) (Icon for an engine subsystem), [VehicleInterfaces.Icons.Empty](#) (Icon for an empty component), [Interfaces.Base](#) (Basic interface definition for an engine).

Parameters

Name	Description
Advanced	
usingMultiBodyAccessories	=true if using MultiBody accessories with a 1D engine
usingMultiBodyTransmission	=true if using a MultiBody transmission with a 1D engine

Connectors

Name	Description
accessoryFlange	Connection for the engine accessories
transmissionFlange	Connection to the transmission
controlBus	Control signal bus
engineMount	Engine mount connection (optional)
acceleratorPedal	Accelerator pedal connection (optional)

VehicleInterfaces.Engines.MinimalEngine

Simple engine model with torque proportional to accelerator pedal position

**Information**

A simple engine model with flywheel and where the torque output is proportional to the accelerator pedal position. This engine model uses a drive-by-wire accelerator, i.e. the accelerator pedal position is read from the control bus.

Extends from [VehicleInterfaces.Icons.Engine](#) (Icon for an engine subsystem), [Interfaces.Base](#) (Basic interface definition for an engine).

Parameters

Name	Description
engineSpeed_start	Initial engine speed [rad/s]
requestedTorque	Engine torque = requested_torque*accelerator_pedal_position [N.m]
flywheelInertia	Flywheel inertia [kg.m2]
axisOfRotation	Axis of rotation of engine resolved in engineMount [1]
Advanced	
usingMultiBodyAccessories	=true if using MultiBody accessories with a 1D engine
usingMultiBodyTransmission	=true if using a MultiBody transmission with a 1D engine

Connectors

Name	Description
accessoryFlange	Connection for the engine accessories
transmissionFlange	Connection to the transmission
controlBus	Control signal bus
engineMount	Engine mount connection (optional)
acceleratorPedal	Accelerator pedal connection (optional)

VehicleInterfaces.Engines.MinimalEngineUsingPedal

Simple engine model with torque proportional to accelerator pedal position, uses accelerator pedal connection



Information

A simple engine model with flywheel and where the torque output is proportional to the accelerator pedal position. This engine model uses the physical connection between the driver and the engine for the accelerator pedal.

Extends from [VehicleInterfaces.Icons.Engine](#) (Icon for an engine subsystem), [Interfaces.Base](#) (Basic interface definition for an engine).

Parameters

Name	Description
engineSpeed_start	Initial engine speed [rad/s]
pedalPositionToTorque	Engine torque = pedalPositionToTorque*accelerator_pedal_position [N]
flywheelInertia	Flywheel inertia [kg.m ²]
axisOfRotation	Axis of rotation resolved in engineMount [1]
Advanced	
usingMultiBodyAccessories	=true if using MultiBody accessories with a 1D engine
usingMultiBodyTransmission	=true if using a MultiBody transmission with a 1D engine

Connectors

Name	Description
accessoryFlange	Connection for the engine accessories
transmissionFlange	Connection to the transmission
controlBus	Control signal bus
engineMount	Engine mount connection (optional)
acceleratorPedal	Accelerator pedal connection (optional)

VehicleInterfaces.PowertrainMounts

Collection of powertrain mounts subsystem definition

Information

The powertrain mounts subsystem interfaces are defined in this sub-package of the VehicleInterfaces library. There are three different powertrain mount interfaces defined and the one to use depends on how many powertrain systems are to be connected to this mounting system. The following connections are provided

- **chassisFrane** - MultiBody connection to transmit the mounting system forces in to the vehicle body
- **mount_n** - MultiBody connection(s) to the powertrain subsystem reference frames

Effects to be modelled in this subsystem

Within the VehicleIntefaces package the powertrain mounts subsystem is used to model how a powertrain subsystem is mounted within the vehicle body. The housing for the powertrain subsystem should be modelled within the relevant powertrain subsystem and this subsystem models the (usually) elastic mounts that suspend the housing within the vehicle body structure.

Extends from [Icons.VariantLibrary](#) (Icon for a package class which contains several variants of one assembly or component).

Package Content

Name	Description
 Tutorial	Powertrain Mounts Tutorial
 Interfaces	Collection of interface definitions for powertrain mounting system
 ThreeSystemRigidMount	3 system rigid mount
 TwoSystemRigidMount	2 system rigid mount
 SingleSystemRigidMount	1 system rigid mount

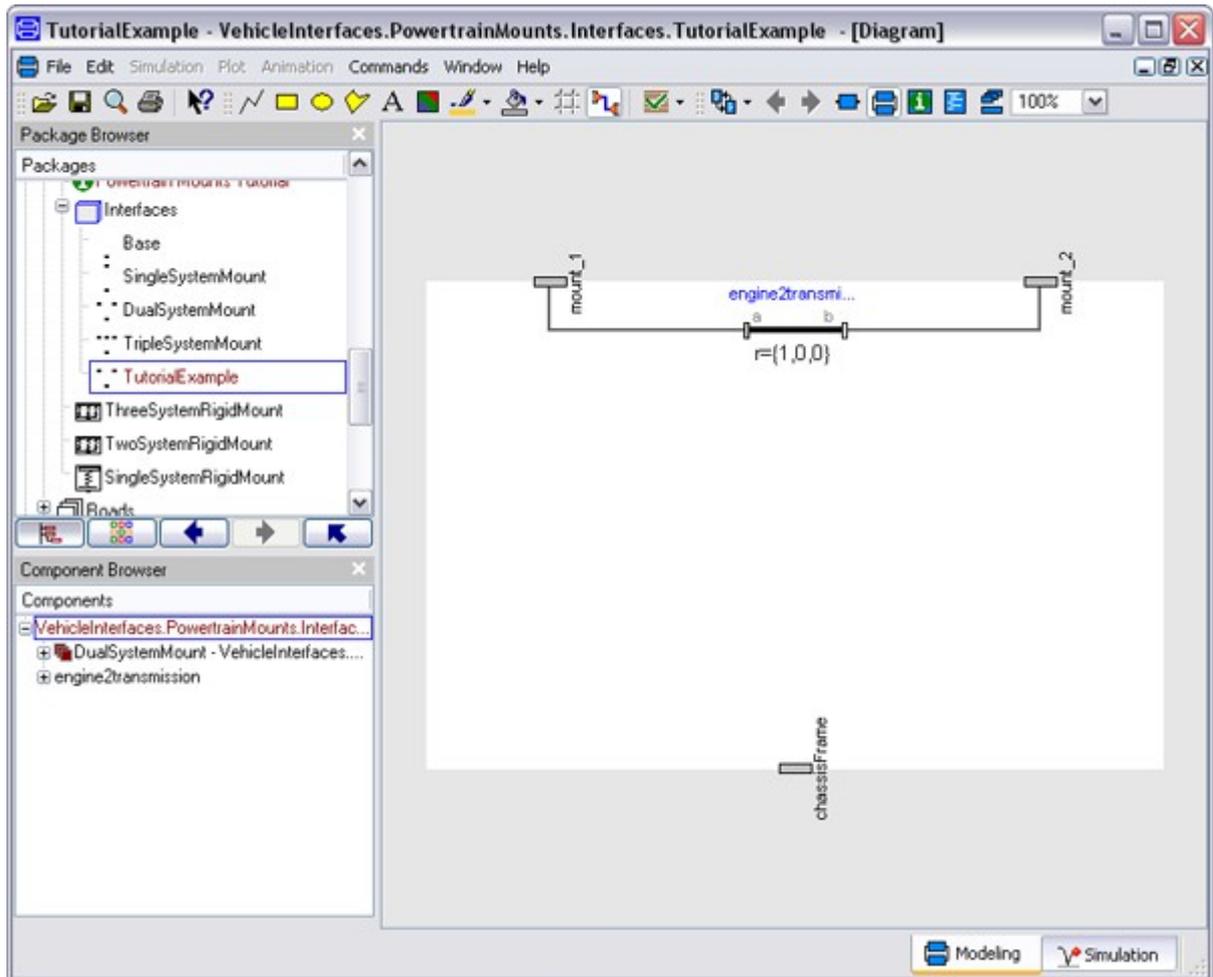
VehicleInterfaces.PowertrainMounts.Tutorial

Tutorial - Defining a new powertrain mounting system

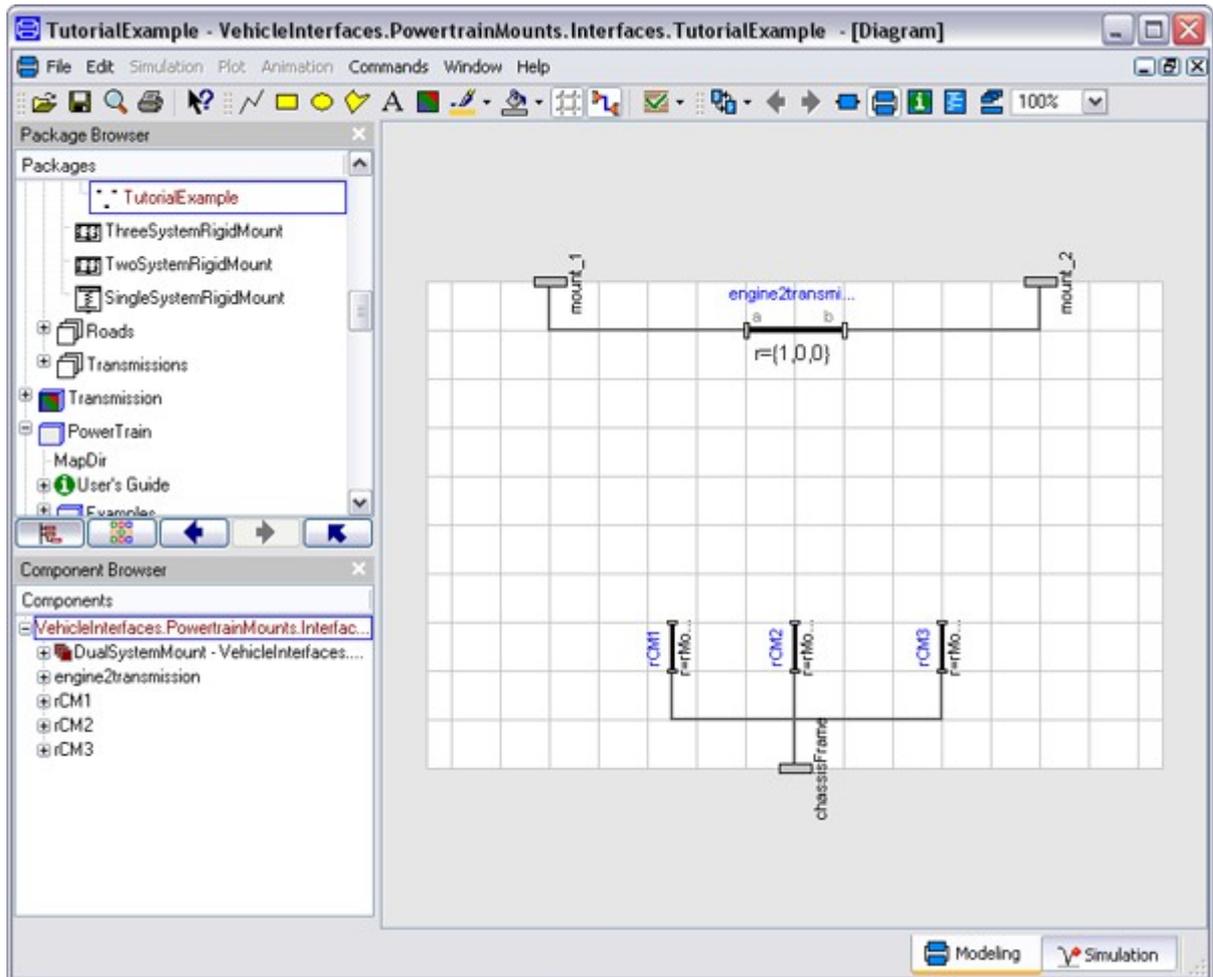
The following process will demonstrate how to create a new powertrain mounting system. In this example we will create a mounting system for the engine and transmission systems where these two systems are rigidly connected together.



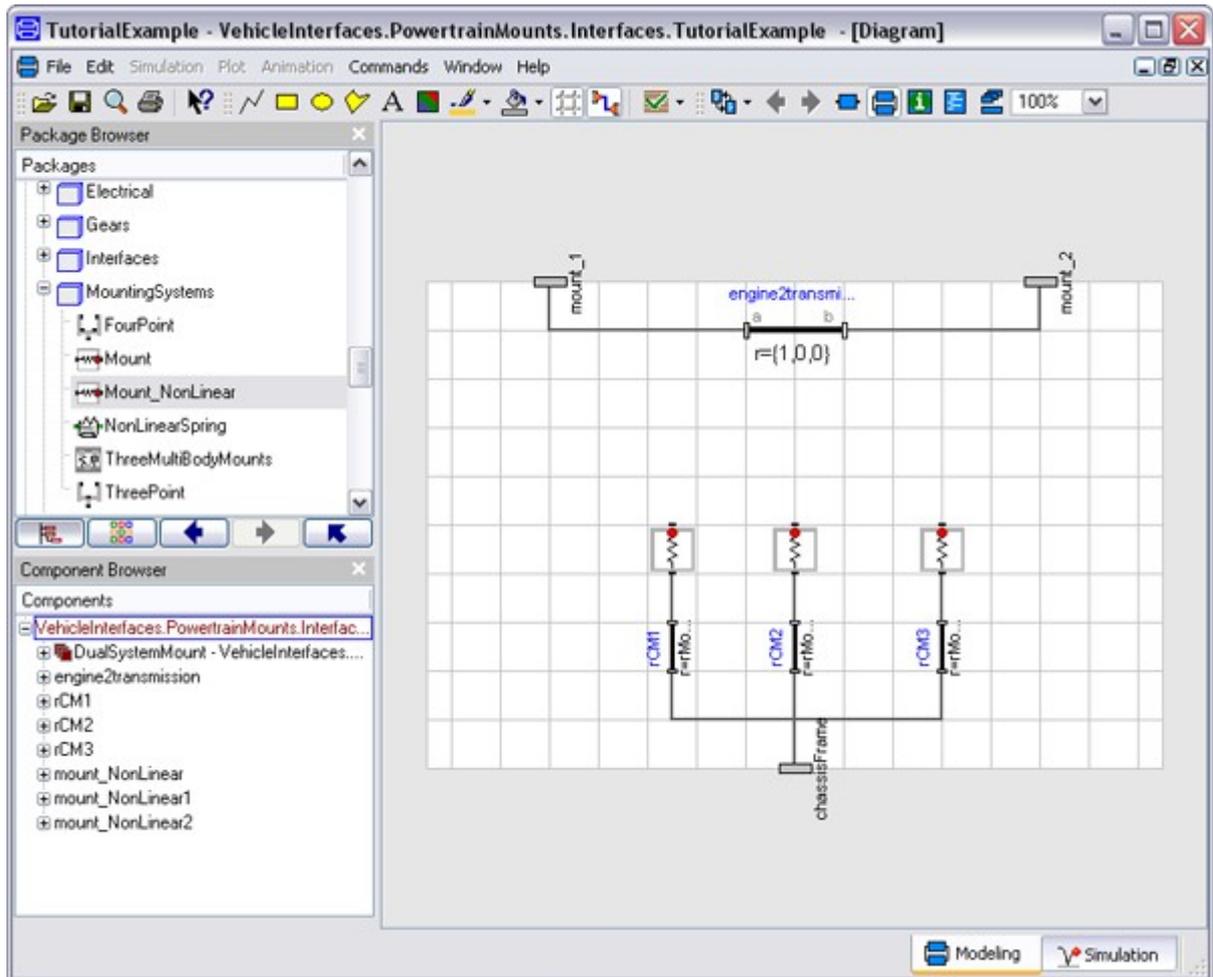
1. We are creating a powertrain mounting system to support 2 powertrain systems so we need to base our model on the **DualSystemMount** Interface definition. Start by extending this class **VehicleInterfaces.PowertrainMounts.Interfaces.DualSystemMount**
2. This interface definition contains 3 MultiBody connectors, the one called **chassisFrame** will be attached to the chassis subsystem in our vehicle model and is the chassis reference frame. The other two connectors, **mount_1** and **mount_2** are the reference frames for the two subsystems that this mounting system will support. In this case **mount_1** will be treated as the engine reference frame and **mount_2** as the transmission reference frame. The first step is to add a fixed translation (**Modelica.Mechanics.MultiBody.Parts.FixedTranslation**) between **mount_1** and **mount_2** to define the position of the transmission reference frame relative to the engine reference frame.



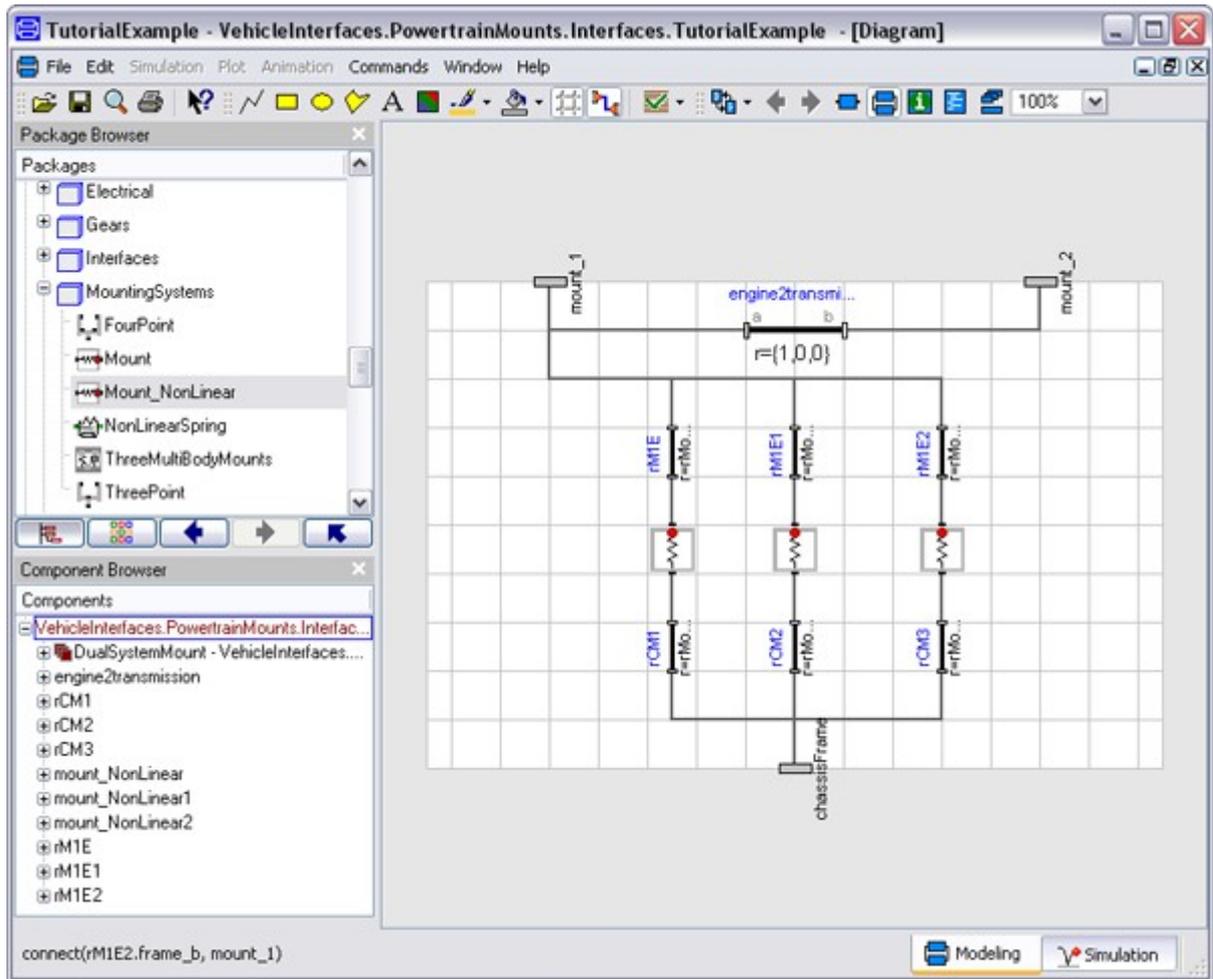
3. The next step is to define the mounting system. In this example we will utilise some components from the PowerTrain library to define the mounting system. The engine and transmission systems will be mounted by 3 elastic mounts. Start defining the mounting system by adding 3 fixed translation blocks that define the locations of the mounts relative to the chassisFrame.



4. Add the elastic mount blocks, in this case they are defined in the PowerTrain library as **PowerTrain.MountingSystems.Mount**



5. Finally add 3 more fixed translation blocks that define the location of the engine reference frame relative to each of the mount locations.



6. The model is now complete and should check correctly

VehicleInterfaces.PowertrainMounts.Interfaces

Collection of interface definitions for powertrain mounting system

Information

A collection of partial base classes which define interfaces for powertrain mounting systems.

Extends from Modelica.Icons.InterfacesPackage (Icon for packages containing interfaces).

Package Content

Name	Description
Base	Basic interface definition for a mounting system
SingleSystemMount	Single powertrain subsystem mounting interface definition
DualSystemMount	Two powertrain subsystem mounting interface definition
TripleSystemMount	Three powertrain subsystem mounting interface definition

VehicleInterfaces.PowertrainMounts.Interfaces.Base**Basic interface definition for a mounting system****Information**

This partial model defines the common interfaces required for a mounting subsystem within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Connectors

Name	Description
chassisFrame	Chassis frame

VehicleInterfaces.PowertrainMounts.Interfaces.SingleSystemMount**Single powertrain subsystem mounting interface definition****Information**

This partial model defines the interfaces required for a single powertrain system mounting subsystem within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Extends from [Base](#) (Basic interface definition for a mounting system).

Connectors

Name	Description
chassisFrame	Chassis frame
mount_1	Powertrain system 1

VehicleInterfaces.PowertrainMounts.Interfaces.DualSystemMount**Two powertrain subsystem mounting interface definition****Information**

This partial model defines the interfaces required for two powertrain system mounting subsystem within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Extends from [Base](#) (Basic interface definition for a mounting system).

Connectors

Name	Description
chassisFrame	Chassis frame
mount_1	Powertrain system 1
mount_2	Powertrain system 2

VehicleInterfaces.PowertrainMounts.Interfaces.TripleSystemMount**Three powertrain subsystem mounting interface definition**

Information

This partial model defines the interfaces required for three powertrain system mounting subsystem within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

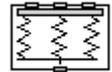
Extends from [Base](#) (Basic interface definition for a mounting system).

Connectors

Name	Description
chassisFrame	Chassis frame
mount_1	Powertrain system 1
mount_2	Powertrain system 2
mount_3	Powertrain system 3

VehicleInterfaces.PowertrainMounts.ThreeSystemRigidMount

3 system rigid mount



Information

Rigidly mounts 3 power train systems such as the engine, transmission and driveline on one supporting system, usually the chassis

Extends from [Interfaces.TripleSystemMount](#) (Three powertrain subsystem mounting interface definition), [VehicleInterfaces.Icons.MultipleMounts](#) (Icon for a multiple mounting subsystem).

Parameters

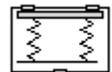
Name	Description
r_ChassisToMount1[3]	Vector from chassis frame to mount_1 frame [m]
r_ChassisToMount2[3]	Vector from chassis frame to mount_2 frame [m]
r_ChassisToMount3[3]	Vector from chassis frame to mount_3 frame [m]

Connectors

Name	Description
chassisFrame	Chassis frame
mount_1	Powertrain system 1
mount_2	Powertrain system 2
mount_3	Powertrain system 3

VehicleInterfaces.PowertrainMounts.TwoSystemRigidMount

2 system rigid mount



Information

Rigidly mounts 2 power train systems such as the engine and transmission on one supporting system, usually the chassis

Extends from [Interfaces.DualSystemMount](#) (Two powertrain subsystem mounting interface definition), [VehicleInterfaces.Icons.TwoMounts](#) (Icon for a multiple mounting subsystem).

Parameters

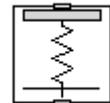
Name	Description
r_ChassisToMount1[3]	Vector from chassis frame to mount_1 frame [m]
r_ChassisToMount2[3]	Vector from chassis frame to mount_2 frame [m]

Connectors

Name	Description
chassisFrame	Chassis frame
mount_1	Powertrain system 1
mount_2	Powertrain system 2

VehicleInterfaces.PowertrainMounts.SingleSystemRigidMount

1 system rigid mount



Information

Rigidly mounts 1 power train systems such as the engine or transmission on one supporting system, usually the chassis

Extends from [Interfaces.SingleSystemMount](#) (Single powertrain subsystem mounting interface definition), [VehicleInterfaces.Icons.SingleMount](#) (Icon for a single mounting subsystem).

Parameters

Name	Description
r_ChassisToMount1[3]	Vector from chassis frame to mount_1 frame [m]

Connectors

Name	Description
chassisFrame	Chassis frame
mount_1	Powertrain system 1

VehicleInterfaces.Roads

Collection of road definitions

Information

The road subsystem interfaces are defined in this sub-package of the VehicleInterfaces library. The road subsystem has no connectors but contains standard functions that can be redeclared to implement different roads. When a road subsystem is used in a model architecture it should be declared as an **inner** system so that it's functions can be accessed by other components in the model.

Effects to be modelled in this subsystem

Within the VehicleInterfaces package the road subsystem is used to define the road surface.

Extends from [VehicleInterfaces.Icons.VariantLibrary](#) (Icon for a package class which contains several variants of one assembly or component).

Package Content

Name	Description
 Tutorial	Roads Tutorial
 Interfaces	Collection of interface definitions for roads
 FlatRoad	Straight road along x-axis (perpendicular to world z-axis)
 CircleRoad	Circular road (perpendicular to world z-axis)

VehicleInterfaces.Roads.Tutorial

Tutorial - Defining a new road

This tutorial will guide you through the steps required to build the FlatRoad model.

1. Create a new road model by extending the base definition which is **VehicleInterfaces.Roads.Interfaces.Base**
2. The minimum steps necessary to define a road model are to provide definitions to the following partial function definitions included in the base road model.
 - **position**
 - **trackOffset**
 - **normal**
 - **headingDirection**
 - **frictionCoefficient**
3. Define the position function. The road is to be defined as a line along the x-axis with the lateral direction along the y-axis. The inputs to the base position function are **s** and **w** and these have the following meaning in this road definition:

```
s: x-coordinate in world frame (x-axis is road heading direction)
w: y-coordinate in world frame (y-axis is lateral displacement)
```

The input arguments **s** and **w** are common to all the road functions that we need to define and are included in the base function definitions. These definitions can then be used to define our position function as follows:

```
function linePosition "Determine point on road"
  extends VehicleInterfaces.Roads.Interfaces.Base.position;
algorithm
  r_0 := {s,w,0};
end linePosition;
```

4. Define the trackOffset function. This road is defined with a constant offset that can be non-zero so the function is defined as:

```
function constantOffset "Determine offset from road centre line"
  extends VehicleInterfaces.Roads.Interfaces.Base.trackOffset;
  input Modelica.SIunits.Distance offset;
algorithm
  trackOffset := {0,offset,0};
end constantOffset;
```

5. Define the normal function. This road is a flat road so the normal is always in the same direction along the z-axis and our normal function can be defined as:



```

function lineNormal "Determine unit normal on road"
  extends VehicleInterfaces.Roads.Interfaces.Base.normal;
algorithm
  e_n_0 := {0,0,1};
end lineNormal;

```

6. Define the headingDirection function. In step 3, we define the road as a line along the x-axis. This means that the heading direction is always in the same direction along the x-axis and our headingDirection function can be defined as:

```

function lineHeadingDirection
  "Determine unit heading direction on road"
  extends VehicleInterfaces.Roads.Interfaces.Base.headingDirection;
algorithm
  e_s_0 := {1,0,0};
end lineHeadingDirection;

```

7. Define the frictionCoefficient function. The friction coefficient could be defined to vary along the road surface but in this case we are going to define it as a constant. Our frictionCoefficient can be defined as:

```

function lineFrictionCoefficient
  "Determine friction coefficient at point on road"
  extends VehicleInterfaces.Roads.Interfaces.Base.frictionCoefficient;
  input Real mue_fixed "Friction coefficient";
algorithm
  mue := mue_fixed;
end lineFrictionCoefficient;

```

8. We now need to redeclare the functions in the base road definition to be the functions shown above. We do this with the following code assuming that the above functions have been defined within the class **FlatRoad**:

```

model FlatRoad "Straight road along x-axis (perpendicular to world z-
axis)"
  extends VehicleInterfaces.Roads.Interfaces.Base(
    redeclare final function position = linePosition,
    redeclare final function trackOffset = constantOffset(offset=offset)
    redeclare final function normal = lineNormal,
    redeclare final function headingDirection = lineHeadingDirection,
    redeclare final function frictionCoefficient =
lineFrictionCoefficient(mue_fixed=mue));

  //rest of definition
  ...
end FlatRoad;

```

9. In redeclaring the functions we have also introduced new parameters **mue** and **offset** that need to be added to the model to define the friction coefficient of the road and the track offset from the road centre line.
10. The **FlatRoad** model included in this package also includes animation of the road surface but this is not essential to the definition of the road and will not be described in this tutorial.
11. The road model is complete and can now be used

VehicleInterfaces.Roads.Interfaces

Collection of interface definitions for roads

Information

Basic interface definition of a generic road as parameterized surface.

Extends from Modelica.Icons.InterfacesPackage (Icon for packages containing interfaces).

Package Content

Name	Description
 positionBase	Determine road position
 trackOffsetBase	Determine track offset from road centre line
 normalBase	Determine unit normal on road
 headingDirectionBase	Determine unit heading direction on road
 frictionCoefficientBase	Determine friction coefficient on road
Base	Base model for all roads

VehicleInterfaces.Roads.Interfaces.positionBase

Determine road position



Information

Partial base model for road position. Extend this model appropriately to define final user model.

Extends from Modelica.Icons.Function (Icon for functions).

Inputs

Name	Description
s	Roads surface parameter 1 (usually arc length along road)
w	Roads surface parameter 2 (usually lateral direction)

Outputs

Name	Description
r_0[3]	Position vector from world frame to point on road at (s,w), resolved in world frame [m]

VehicleInterfaces.Roads.Interfaces.trackOffsetBase

Determine track offset from road centre line



Information

Partial base model for track offset from road centre line. Extend this model appropriately to define final user model.

Extends from Modelica.Icons.Function (Icon for functions).

Inputs

Name	Description
s	Roads surface parameter 1 (usually arc length along road)
w	Roads surface parameter 2 (usually lateral direction)

Outputs

Name	Description
trackOffset[3]	Track offset vector from road centre line to desired trajectory point [m]

VehicleInterfaces.Roads.Interfaces.normalBase**Determine unit normal on road****Information**

Partial base model for unit normal on road. Extend this model appropriately to define final user model.

Extends from Modelica.Icons.Function (Icon for functions).

Inputs

Name	Description
s	Roads surface parameter 1 (usually arc length along road)
w	Roads surface parameter 2 (usually lateral direction)

Outputs

Name	Description
e_n_0[3]	Unit normal of road at (s,w), resolved in world frame

VehicleInterfaces.Roads.Interfaces.headingDirectionBase**Determine unit heading direction on road****Information**

Partial base model for unit heading direction on road. Extend this model appropriately to define final user model.

Extends from Modelica.Icons.Function (Icon for functions).

Inputs

Name	Description
s	Roads surface parameter 1 (usually arc length along road)
w	Roads surface parameter 2 (usually lateral direction)

Outputs

Name	Description
e_s_0[3]	Unit vector in direction of road heading at (s,w), resolved in world frame

VehicleInterfaces.Roads.Interfaces.frictionCoefficientBase

Determine friction coefficient on road

**Information**

Partial base model for friction coefficient at point on road. Extend this model appropriately to define final user model.

Extends from Modelica.Icons.Function (Icon for functions).

Inputs

Name	Description
s	Roads surface parameter 1 (usually arc length along road)
w	Roads surface parameter 2 (usually lateral direction)

Outputs

Name	Description
mue	Roads friction coefficient at (s,w)

VehicleInterfaces.Roads.Interfaces.Base

Base model for all roads

Information

A vehicle is driving on a road that is described as parameterized surface $r_0 = r_0(s, w)$, where (s, w) are the surface parameters. Usually, "s" describes the heading direction, i.e., the direction in which the vehicle is driving, whereas "w" describes the lateral direction of the track.

VehicleInterfaces.Roads.FlatRoad

Straight road along x-axis (perpendicular to world z-axis)

**Information**

The road is a line along the x-axis of the world system. The road surface parameters are defined as:

s: x-coordinate in world frame (x-axis is road heading direction)
 w: y-coordinate in world frame (y-axis is lateral displacement)

The point s=0, w=0 has position vector $r=\{0, 0, 0\}$;

Extends from [VehicleInterfaces.Icons.Road](#) (Icon for a road), [VehicleInterfaces.Roads.Interfaces.Base](#) (Base model for all roads).

Parameters

Name	Description
replaceable function position	Dummy model for road position
replaceable function trackOffset	Dummy model for track offset from road centre line
replaceable function normal	Dummy model for unit normal on road

replaceable function headingDirection	Dummy model for unit heading direction on road
replaceable function frictionCoefficient	Dummy model for friction coefficient on road
animation	= true, if road shall be visualized
mue	Friction coefficient of road
offset	Offset from the road centre line [m]
Animation	
roadColor	Color of road
width	Width of road [m]
x_min	Roads is visualized in the range [x_min .. x_max] [m]
x_max	Roads is visualized in the range [x_min .. x_max] [m]

Connectors

Name	Description
replaceable function position	Dummy model for road position
replaceable function trackOffset	Dummy model for track offset from road centre line
replaceable function normal	Dummy model for unit normal on road
replaceable function headingDirection	Dummy model for unit heading direction on road
replaceable function frictionCoefficient	Dummy model for friction coefficient on road

VehicleInterfaces.Roads.CircleRoad

Circular road (perpendicular to world z-axis)



Information

The road is a circle described by "radius" and "width". The road surface parameters are defined as:

```
s: arc length along circle
w: lateral displacement of road, perpendicular to circle
```

The point $s=0, w=0$ has position vector $r=\{0, -\text{radius}, 0\}$;

The functions are defined in such a way that Dymola inlines all the functions. As a result, it is easy to get derivatives of the functions.

Extends from [VehicleInterfaces.Roads.Interfaces.Base](#) (Base model for all roads).

Parameters

Name	Description
replaceable function position	Dummy model for road position
replaceable function trackOffset	Dummy model for track offset from road centre line
replaceable function normal	Dummy model for unit normal on road
replaceable function headingDirection	Dummy model for unit heading direction on road
replaceable function frictionCoefficient	Dummy model for friction coefficient on road
animation	= true, if road shall be visualized
radius	Radius of road [m]
width	Width of road [m]
mue	Friction coefficient of road

roadColor	Color of road
-----------	---------------

Connectors

Name	Description
replaceable function position	Dummy model for road position
replaceable function trackOffset	Dummy model for track offset from road centre line
replaceable function normal	Dummy model for unit normal on road
replaceable function headingDirection	Dummy model for unit heading direction on road
replaceable function frictionCoefficient	Dummy model for friction coefficient on road

VehicleInterfaces.Transmissions

Collection of transmission subsystem definitions

Information

The transmission subsystem interfaces are defined in this sub-package of the VehicleInterfaces library. The transmission subsystem has the following connectors some of which are optional (see below for more information):

- **engineFlange** - 1D rotational connection to the engine subsystem (or other systems connected to the transmission input)
- **drivelineFlange** - 1D rotational connection to the driveline subsystem (or other systems connected to the transmission output)
- **controlBus** - control signal bus connection
- **clutchPedal** - 1D translational connection for the clutch pedal connection to the driverEnvironment (optional, for manual gearboxes only)
- **shiftConnector** - shift connection to the driverEnvironment (optional, for manual gearboxes only)
- **transmissionMount** - MultiBody connection to transmit the transmission mount reactions (optional)

The optional connectors are, by default, disabled and can be ignored if not required. They can be enabled by setting the appropriate parameter to be true. This is only possible at design time, i.e. when you are building the subsystem model.

Effects to be modelled in this subsystem

Within the VehicleInterfaces package the transmission subsystem is used to model the transmission of torque from the input shaft to the output shaft. The connections to the engine and driveline subsystems are via 1D rotational connectors. The torque reaction in to the transmission housing and the housing itself are also to be modelled in this subsystem if required. The torque reactions, if included, should all be referred back to a single reference frame for the transmission housing (the transmissionMount connector).

Extends from [VehicleInterfaces.Icons.VariantLibrary](#) (Icon for a package class which contains several variants of one assembly or component).

Package Content

Name	Description
 Tutorial	Transmissions Tutorial
 Interfaces	Collection of interface definitions for transmission
 NoTransmissions	Empty transmission
 SingleGearAutomaticTransmission	Simple fixed gear ratio, automatic transmission, no torque converter

 SingleGearManualTransmission	Simple fixed gear ratio, manual transmission, uses physical connectors
 PowerSplitDevice	Simple power split device based on an ideal epicyclic gear

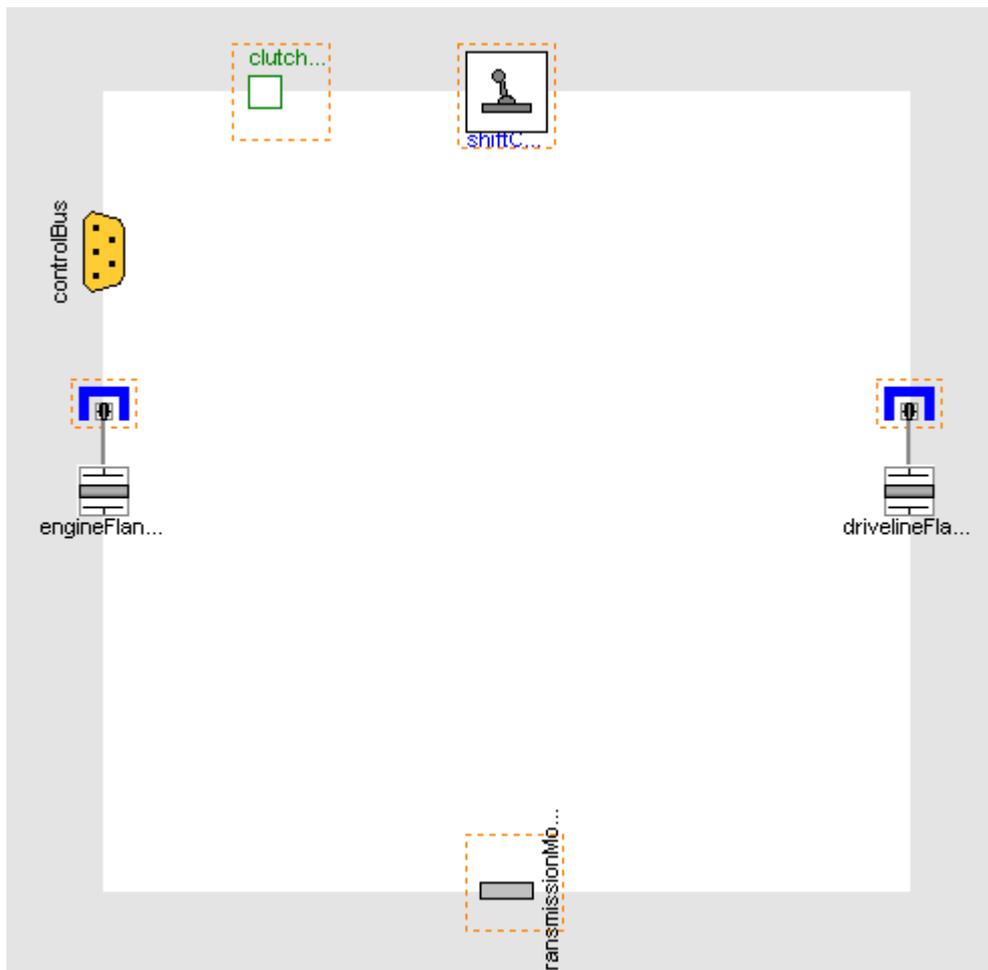
VehicleInterfaces.Transmissions.Tutorial

Tutorial - Defining a new manual transmission model

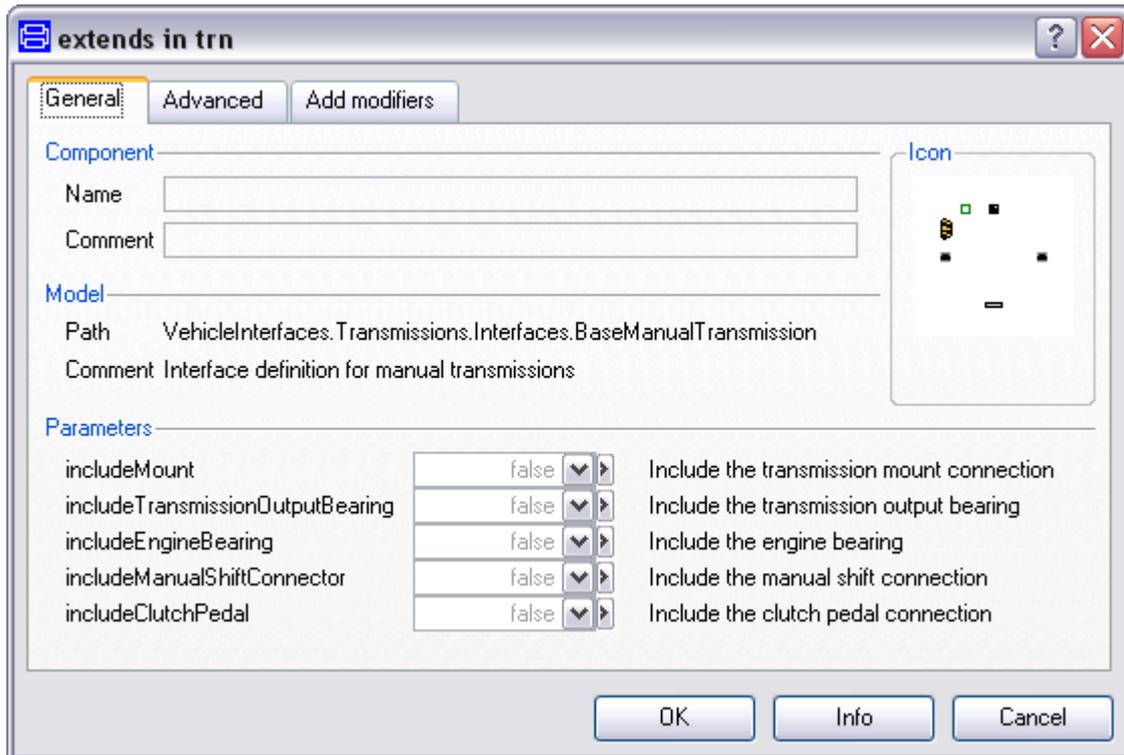
The following process will demonstrate how to create a new manual transmission model using this interface definition.



1. Create a new model that extends **VehicleInterfaces.Transmissions.Interfaces.BaseManualTransmission**, it should look like this:



2. In the component browser, right click on **Base** and select **Parameters** from the context menu to produce the following parameter dialog



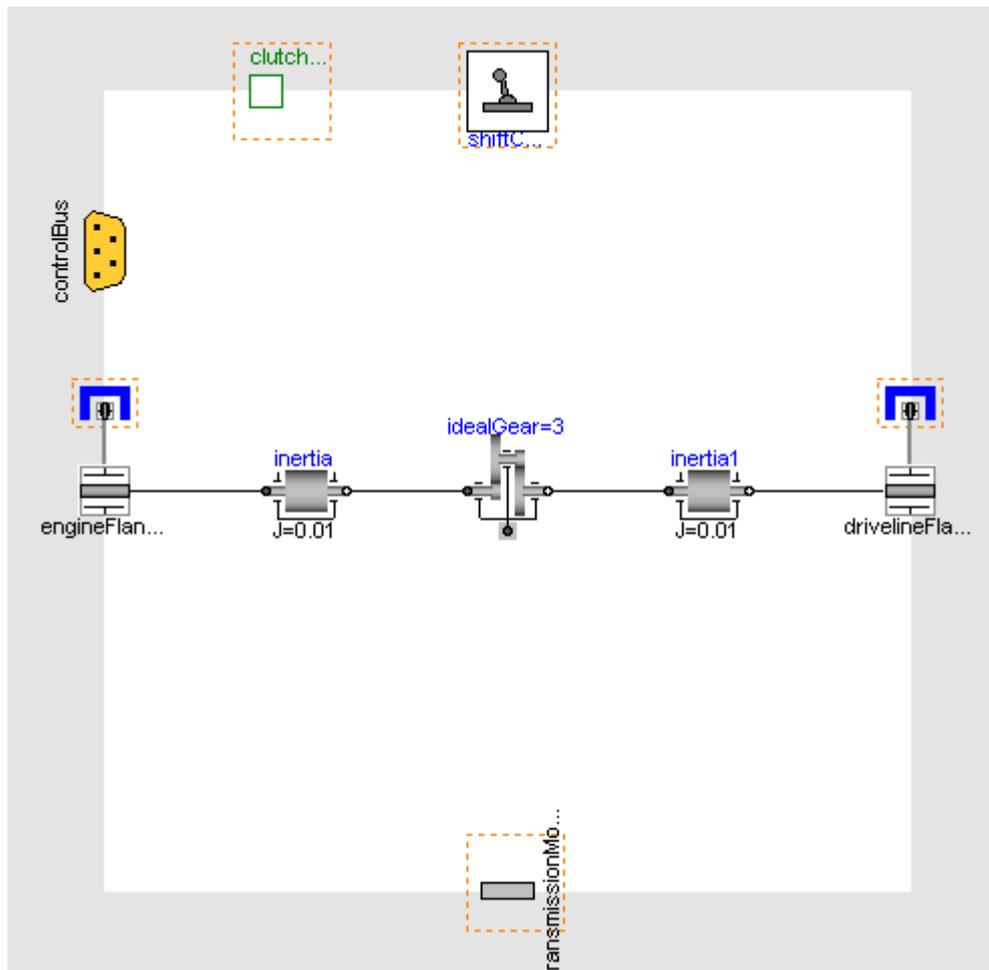
3. This dialog allows you to enable/disable the optional connections by setting **includeClutchPedal**, **includeManualShiftConnector** and **includeMount** as required for your new transmission model. The **engineFlange** and **drivelineFlange** connectors are of the type [VehicleInterfaces.Interfaces.FlangeWithBearing](#), the parameters **includeTransmissionBearing** and **includeDrivelineBearing** controls whether the bearing connectors within these connections are enabled or not.
4. You can now define your transmission model as required

Creating the MinimalTransmission example

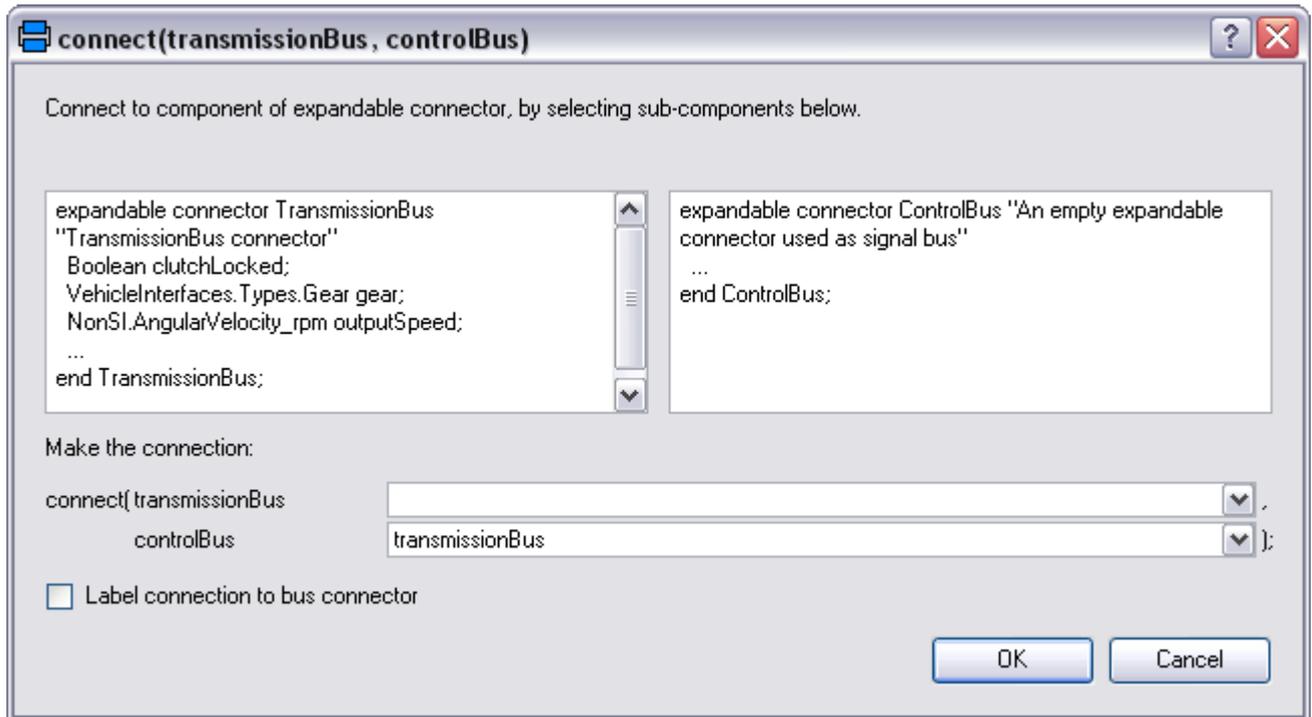
The following steps demonstrate how to create a basic transmission model. The transmission model will consist of a fixed single ratio between the input and output shafts. No torque reaction in to the transmission housing will be modelled.

Starting from step 3 above.

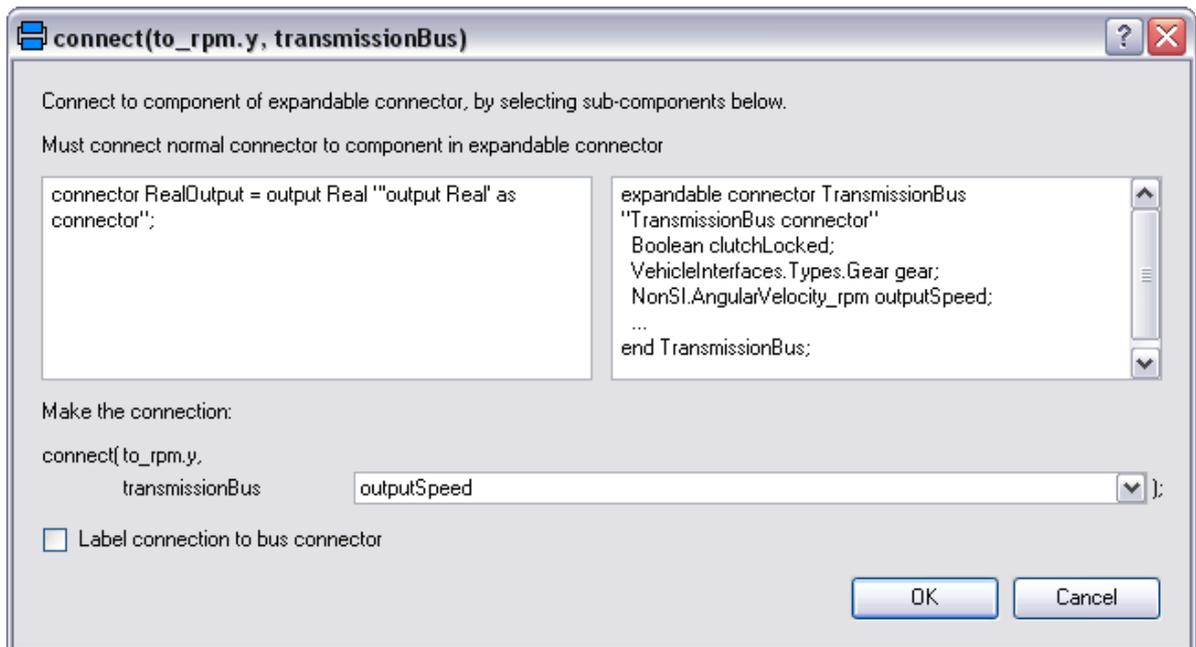
1. First, decide which of the optional connectors are required to model. For this example we don't need any of the optional connections
2. Add the following blocks and connections to the diagram



- Next, we need to add the required connections to the control signal bus for the transmission, see [here](#) for a complete list of the minimum connections required. As we are creating a manual transmission model we need to add three signals to the transmissionBus which is part of the controlBus. We need to put the transmission output speed, current gear and the clutch state on to the transmissionBus. As this is a simple single gear transmission the current gear and clutch state can be set as constants. Start by adding the transmissionBus connector which can be found at **VehicleInterfaces.Interfaces.TransmissionBus** and connect this to the controlBus. When this connection is made the following dialog will be produced and should be completed as shown.

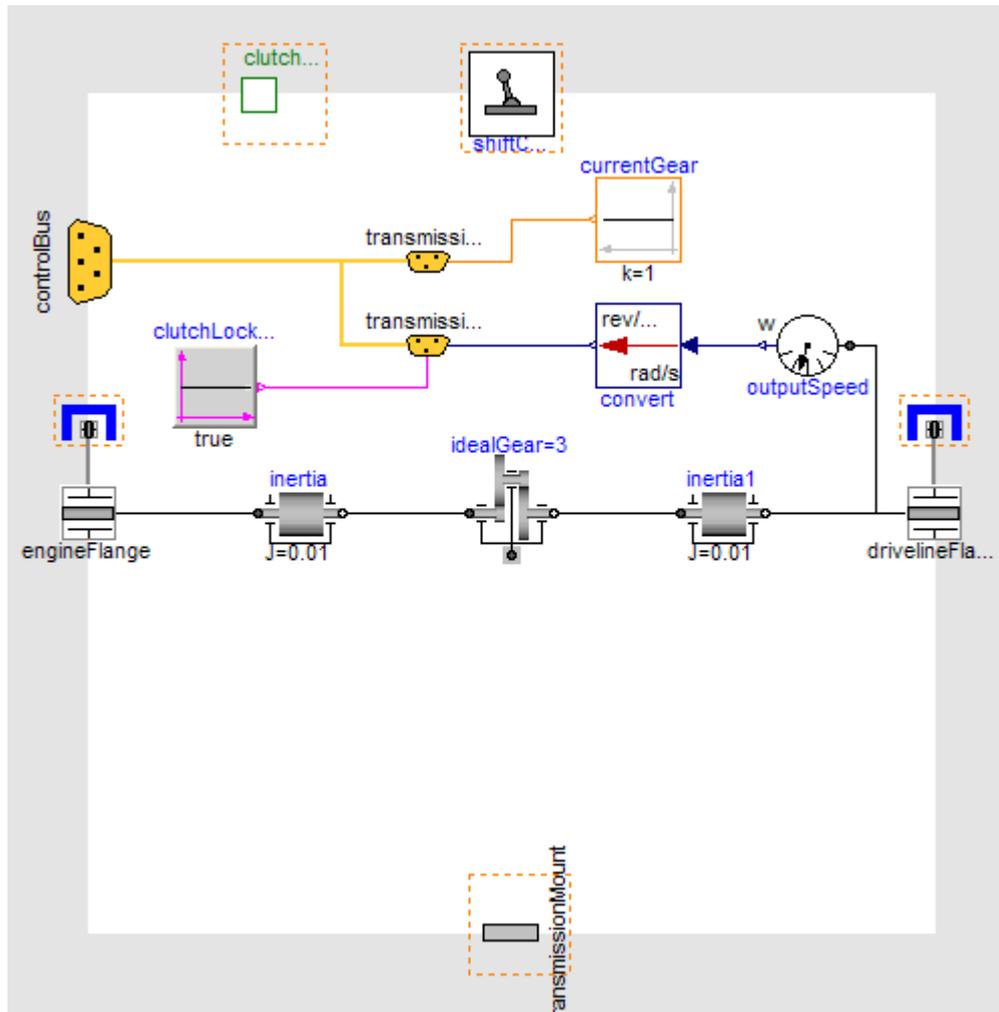


4. We can now connect the constants for clutch state and current gear to the transmissionBus. The transmission output speed needs to be in the correct units, in this case r/min (or rpm). Add a rotational speed sensor and a unit conversion block from **Modelica.Blocks.Math.UnitConversions**, set the conversion block to convert to rpm. When you create the connection between the transmissionBus and one of these blocks a dialog like the one below will be produced. You will need to complete the dialog using one of the following names:
- gear
 - clutchLocked
 - outputSpeed



5. The model is now complete and should check successfully and can be used in any model compatible

with the VehicleInterfaces library



VehicleInterfaces.Transmissions.Interfaces

Collection of interface definitions for transmission

Information

A collection of partial base classes which define interfaces for transmission models.

Extends from Modelica.Icons.InterfacesPackage (Icon for packages containing interfaces).

Package Content

Name	Description
. Base	Basic interface definition for a transmission
. BaseAutomaticTransmission	Interface definition for an automatic transmission
. BaseManualTransmission	Interface definition for a manual transmission
. BaseTwoInputTransmission	Interface definition for a transmission with two input shafts
StandardBus	Bus of VehicleInterfaces.Transmission: StandardBus of signals generated

	by the Transmission model
 StandardControlBus	Bus of VehicleInterfaces.Transmission: StandardControlBus of signals generated by the Transmission Controller model

VehicleInterfaces.Transmissions.Interfaces.Base

Basic interface definition for a transmission

Information

This partial model defines the common interfaces required for a transmission subsystem within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Parameters

Name	Description
Advanced	
usingMultiBodyEngine	=true if using MultiBody engine with a 1D transmission
usingMultiBodyDriveline	=true if using a MultiBody driveline with a 1D transmission

Connectors

Name	Description
engineFlange	Connection to the engine
drivelineFlange	Connection to the driveline
controlBus	Control signal bus
transmissionMount	Transmission mount connection (optional)

VehicleInterfaces.Transmissions.Interfaces.BaseAutomaticTransmission

Interface definition for an automatic transmission

Information

This partial model defines the interfaces required for an automatic transmission model within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Extends from [Base](#) (Basic interface definition for a transmission).

Parameters

Name	Description
Advanced	
usingMultiBodyEngine	=true if using MultiBody engine with a 1D transmission
usingMultiBodyDriveline	=true if using a MultiBody driveline with a 1D transmission

Connectors

Name	Description
engineFlange	Connection to the engine
drivelineFlange	Connection to the driveline

controlBus	Control signal bus
transmissionMount	Transmission mount connection (optional)

VehicleInterfaces.Transmissions.Interfaces.BaseManualTransmission

Interface definition for a manual transmission



Information

This partial model defines the interfaces required for a manual transmission model within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Extends from [Base](#) (Basic interface definition for a transmission).

Parameters

Name	Description
Advanced	
usingMultiBodyEngine	=true if using MultiBody engine with a 1D transmission
usingMultiBodyDriveline	=true if using a MultiBody driveline with a 1D transmission

Connectors

Name	Description
engineFlange	Connection to the engine
drivelineFlange	Connection to the driveline
controlBus	Control signal bus
transmissionMount	Transmission mount connection (optional)
clutchPedal	Clutch pedal connection (optional)
shiftConnector	Manual shift selector connection (optional)

VehicleInterfaces.Transmissions.Interfaces.BaseTwoInputTransmission

Interface definition for a transmission with two input shafts



Information

This partial model defines the interfaces required for a transmission model that has two input shafts (such as a power-split device) within the VehicleInterfaces package. See the [documentation](#) and [tutorial](#) for more information.

Extends from [Base](#) (Basic interface definition for a transmission).

Parameters

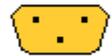
Name	Description
Advanced	
usingMultiBodyEngine	=true if using MultiBody engine with a 1D transmission
usingMultiBodyDriveline	=true if using a MultiBody driveline with a 1D transmission
usingMultiBodyMotor	=true if using MultiBody motor with a 1D transmission

Connectors

Name	Description
engineFlange	Connection to the engine
drivelineFlange	Connection to the driveline
controlBus	Control signal bus
transmissionMount	Transmission mount connection (optional)
motorFlange	Connection to the engine

VehicleInterfaces.Transmissions.Interfaces.StandardBus

Bus of VehicleInterfaces.Transmission: StandardBus of signals generated by the Transmission model



Information

Bus with a set of standard signals generated by the transmission subsystem.

Extends from [.VehicleInterfaces.Interfaces.TransmissionBus](#) (Bus of VehicleInterfaces.Interfaces: Empty expandable connector used as transmission bus), [.VehicleInterfaces.Icons.SignalSubBusWithExplicitSignals](#) (Icon for signal sub-bus where the explicit signals are defined in the bus).

Parameters

Name	Description
outputSpeed	Output shaft speed [rad/s]
clutchLocked	Clutch state (true if locked)

Contents

Name	Description
outputSpeed	Output shaft speed [rad/s]
clutchLocked	Clutch state (true if locked)

VehicleInterfaces.Transmissions.Interfaces.StandardControlBus

Bus of VehicleInterfaces.Transmission: StandardControlBus of signals generated by the Transmission Controller model



Information

Bus with a set of standard signals generated by the controller for a transmission subsystem.

Extends from [.VehicleInterfaces.Interfaces.TransmissionControlBus](#) (Bus of VehicleInterfaces.Interfaces: Bus that contains a minimal set of signals generated by the Transmission Controller model), [.VehicleInterfaces.Icons.SignalSubBusWithExplicitSignals](#) (Icon for signal sub-bus where the explicit signals are defined in the bus).

Parameters

Name	Description
currentGear	Currently selected gear

Contents

Name	Description
currentGear	Currently selected gear

VehicleInterfaces.Transmissions.NoTransmissions

Empty transmission



Information

Empty transmission model (just rigid connection between engine and driveline flange). Using this empty model in overall vehicle architecture the functionality of transmission can be eliminated.

Extends from [VehicleInterfaces.Icons.Transmission](#) (Icon for a transmission subsystem), [VehicleInterfaces.Icons.Empty](#) (Icon for an empty component), [Interfaces.Base](#) (Basic interface definition for a transmission).

Parameters

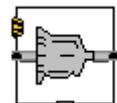
Name	Description
Advanced	
usingMultiBodyEngine	=true if using MultiBody engine with a 1D transmission
usingMultiBodyDriveline	=true if using a MultiBody driveline with a 1D transmission

Connectors

Name	Description
engineFlange	Connection to the engine
drivelineFlange	Connection to the driveline
controlBus	Control signal bus
transmissionMount	Transmission mount connection (optional)

VehicleInterfaces.Transmissions.SingleGearAutomaticTransmission

Simple fixed gear ratio, automatic transmission, no torque converter



Information

A single gear transmission without a launch device that is based on the automatic transmission model interface definition

Extends from [VehicleInterfaces.Icons.Transmission](#) (Icon for a transmission subsystem), [Interfaces.BaseAutomaticTransmission](#) (Interface definition for an automatic transmission).

Parameters

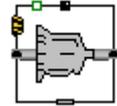
Name	Description
gearRatio	Gear ratio
Advanced	
usingMultiBodyEngine	=true if using MultiBody engine with a 1D transmission
usingMultiBodyDriveline	=true if using a MultiBody driveline with a 1D transmission

Connectors

Name	Description
engineFlange	Connection to the engine
drivelineFlange	Connection to the driveline
controlBus	Control signal bus
transmissionMount	Transmission mount connection (optional)

VehicleInterfaces.Transmissions.SingleGearManualTransmission

Simple fixed gear ratio, manual transmission, uses physical connectors



Information

A single gear transmission based on the manual transmission model interface definition. Includes a clutch model and uses physical connections between the driver and transmission system for the clutch position and current gear number (although the gear number is ignored in this model).

Extends from [VehicleInterfaces.Icons.Transmission](#) (Icon for a transmission subsystem), [Interfaces.BaseManualTransmission](#) (Interface definition for a manual transmission).

Parameters

Name	Description
gearRatio	Gear ratio
Advanced	
usingMultiBodyEngine	=true if using MultiBody engine with a 1D transmission
usingMultiBodyDriveline	=true if using a MultiBody driveline with a 1D transmission

Connectors

Name	Description
engineFlange	Connection to the engine
drivelineFlange	Connection to the driveline
controlBus	Control signal bus
transmissionMount	Transmission mount connection (optional)
clutchPedal	Clutch pedal connection (optional)
shiftConnector	Manual shift selector connection (optional)

VehicleInterfaces.Transmissions.PowerSplitDevice

Simple power split device based on an ideal epicyclic gear



Information

Simple power-split device based on an ideal epicyclic gear. No losses are included in this model

Extends from [Interfaces.BaseTwoInputTransmission](#) (Interface definition for a transmission with two input shafts).

Parameters

Name	Description
ratio	Number of ring_teeth/sun_teeth (e.g. ratio=100/50)
Advanced	
usingMultiBodyEngine	=true if using MultiBody engine with a 1D transmission
usingMultiBodyDriveline	=true if using a MultiBody driveline with a 1D transmission
usingMultiBodyMotor	=true if using MultiBody motor with a 1D transmission

Connectors

Name	Description
engineFlange	Connection to the engine
drivelineFlange	Connection to the driveline
controlBus	Control signal bus
transmissionMount	Transmission mount connection (optional)
motorFlange	Connection to the engine