755 Phoenix Drive, Ann Arbor MI, 48108, USA
Phone: 734 668-2930 • Fax: 734 668-2877 • Email: info@carsim.com

CarSim New Features

This document lists notable new features in CarSim 2024.3 and other releases going back to 2021.1.

CarSim 2024.3 New Features

VS Math Models

Dynamic Suspension Compliance on X and Y Degrees of Freedom

In this release, a new independent suspension module is supported, which enables dynamic compliance for both X and Y degrees of freedom (DOFs) at the same time. This module can be activated by a new VS Command `DEFINE_SUSP_XY_DOF`. Note that there is currently not a corresponding checkbox in CarSim GUI screen to activate this module, and the users must specify the VS Command in a miscellaneous yellow field; see the *Suspension Systems* help manual for details of the usage. Recall that as of version 2024.2, CarSim independent suspensions support dynamic compliance only on the X DOF with the VS Command `DEFINE_SUSP_X_DOF`, and a corresponding checkbox exists on the GUI screen Suspension: Compliance (Nonlinear). These functionalities remain as before, allowing continuing users to keep using X-DOF-only dynamic compliance.

Shift Option for Parallel Hybrid Powertrain

A new parameter `OPT_HEV_SHIFT_COMMAND` is added to specify the throttle command used for the transmission shift from either HEV controller (=0, default) or driver pedal (= 1.) This option is only available for the *pre*-transmission motor case of the parallel hybrid powertrain system.

Simulink S-Function

Starting with version 2024.3, we support Simulink S-Function with bus I/O. Each bus signal is named by the short name of the solver. We added two new S-Function blocks:

1. Input is vector, output is bus

2. Both input and output are buses.

VS Browser: Graphical User Interface (GUI)

1. Updated powertrain differential screens (front, rear, transfer case) to provide better documentation when right-clicking on yellow fields, and generate Parsfiles without extra copies of the data for very old versions of the software (more than 15 years old).

Real-Time

A&D

VS Connect is now supported on A&D HELIOS platform

Documentation

The following documents were added:

1. Technical Memos > Dynamic Suspension Compliance

The following Guides were updated:

2. Running a VS Math Model in Simulink

The following documents in the **Help** menu were updated:

3. Paths, Road Surfaces, and Scenes > VS Terrain
4. Steering > Steering Systems
5. Suspension Systems
6. Tires > Tire Models

The following RT and DS documents were updated:

7. dSPACE Guide

Database

Examples were added or updated in the following categories.

Parallel Hybrid Shift Option

Two new examples are added for a comparison between the transmission shift by HEV controller and driver pedal (distinguished by a new parameter `OPT_HEV_SHIFT_COMMAND`) in the case of *pre*-transmission motor of the parallel hybrid powertrain system.

XY DOF: Curb Impact

This is a new category consisting of four new curb-striking examples, under four different suspension compliance and tire model settings:

- X and Y DOF dynamic compliance; MF-Swift tire with enveloping feature.
- X DOF dynamic and Y DOF quasi-static compliance; MF-Swift tire with enveloping feature.
- X and Y DOF dynamic compliance; extended internal table lookup tire.

- X DOF dynamic and Y DOF quasi-static compliance; extended internal table lookup tire.

XY DOF with External Tire Models

This is a new category consisting of two new examples of co-simulation with COSIN FTire and Fraunhofer CDTire under large suspension compliances. Dynamic compliances were enabled for independent suspensions on both X and Y DOFs.

Simulink Bus Signals for VehicleSim S-Function

A new example, Ext. ABS: Split Mu -Bus, is added to the Simulink Models category to demonstrate the Simulink S-Function with bus I/O. This block utilizes Simulink bus objects and incorporates signal names directly into Simulink.

Refined EV/HEV example datasets (several categories)

All electric and hybrid electric powertrain example datasets were reviewed and refined for faster acceleration based on the following aspects:

- Added high powered motor datasets for full electric vehicles.
- Total motor (+ engine) power and maximum driver require power (`PWR_DRV_THROTTLE_COEFFICIENT`) are matched.
- Lower the discharge/charging resistances to 4-5 milliohms for higher battery power limit.
- Set higher battery voltage (increased from 400 to 800V) especially for the vehicle with multiple motors.

CarSim 2024.2 New Features

VS Math Models

VS STI Callback Functions from VS Solvers

Various VS Solvers functions are available for use in the VS STI module as callbacks. In the past, any tire model using VS STI module required all callback functions to be defined. In this release, all these functions are optional for a tire model, i.e., a tire model needs to define them only if they are used in the specific model.

VS STI Module for Fraunhofer CDTire

A VS STI module for the Fraunhofer CDTire is included for this release. This interface is compatible with CDTire version 2023.1.1 and it enables to co-simulate with CDTire model on Windows 64-bit.

VS Embedded Python

The Embedded Python versions of VS API functions to get and set values for variables in the VS Math Model worked with most types of variables that have keyword names, such as parameters, state variables, output variables, and miscellaneous named variables. The Embedded Python versions now also work with import variables (those with keywords of the form `IMP_`*name*).

External Table Files

About half of the calculations made in VS Math Models involve tables that are used to calculate a dependent variable from one or two known variables. In most cases, tabular data from test rigs or other software are routinely put into the VS Browser via copy/paste. However, there are occasions where simple copy/paste into VS Browser table screens is inefficient. Alternative methods are available to provide tables with three types of external data files:

1. Parsfiles added to the model with an `INCLUDE` statement.
2. Binary VSTB files made from a VS Math Model or a VS utility program.
3. Output VS or ERD files.

These capabilities are supported with new examples, new documentation, and improvements in the VSTB file option. A new Tech Memo describes some of the examples and documents a new utility `vs_csv_to_vstb` that creates VSTB files from spreadsheet CSV files.

Regulating Hybrid/Electric Powertrain Motor Torque under Low Speed

New parameter (`VLOW_POWER_TRQ`) has been added in order to regulate motor demand torque at low speed. The motor demand torque used to be calculated by the power demand from driver divided by either transmission output speed or a magic number of 10 rad/sec whichever the larger value. With such calculation, the motor demand torque could be very large value when the vehicle speed is very low. In order to regulate the large demand torque at low speed, the magic number is replaced by the transmission speed derived from the new parameter. Setting `-1` to the new parameter provides the same behavior as prior versions for backward compatibility.

Real-Time

Speedgoat

Running VehicleSim FMU2.0 is now supported on Speedgoat target.

NI VeriStand LinuxRT

Support Simulink S-Function “vs_sf” running on NI VeriStand Linux RT system.

ETAS

Running VehicleSim on ETAS target is no longer supported.

VS Terrain

The VS Terrain Utility and Library introduces version 3 of the format, which has been reworked to build terrain files much faster, while handling much larger scenes. The api allows for geometry to be added in bulk, in a format used in the most common file types. There is also a new V3 Virtual build type that can position many terrain files within a virtual space using the new .vsatlas file format.

Documentation

The following documents were added:

4. Reference Manuals > VS Embedded Python Interface
5. Technical Memos > Options for Loading Table Data into VS Math Models
6. Technical Memos > VS Embedded Python Interface Examples

The following VS Reference Manuals were updated:

7. System Parameters in VS Math Models
8. VS Commands
9. VS Math Models
10. VS Output API
11. VS Visualizer

The following documents in the **Help** menu were updated:

12. Controls > Driver Controls
13. Model Extensions and RT > Custom Forces and Motion Sensors
14. Paths, Road Surfaces, and Scenes > Paths and Road Surfaces
15. Paths, Road Surfaces, and Scenes > VS Terrain
16. Powertrain > Electric and Hybrid-Electric System (BEV/HEV)
17. Tires > Tire Models

The following Technical Memos were updated:

18. Connecting to External Tire Models with VS STI

The following RT and DS documents were updated:

19. A&D RT Guide
20. Concurrent RT Guide
21. NI RT Guide
22. Speedgoat Guide for VehicleSim Products

The following VS SDK documents were updated:

23. Extending a Model with VS Commands and the API

Database

Examples were added or updated in the following categories.

Embedded Python

Nearly all existing examples for Embedded Python were replaced, and new examples were added to the same category. The category has a '*' prefix for this release.

External Table Files

Example runs show use of three kinds of external table files: Parsfiles loaded with the INCLUDE statement, binary VSTB files made with the VS Math Model or the new vs_csv_to_vstb tool, and tables made with data from output VS or ERD files.

SDK: Extended Models

The example path following steering controller was updated to remove some complexity dealing with initialization that was needed for older versions of the models (prior to 2023). The examples are now simpler and avoid a minor discontinuity at the start.

CDTire (External software)

Five new examples of co-simulation with Fraunhofer CDTire were added. The examples involved CDTire-3D and CDTire-RT with and without parallel processing option. Sample CDTire data files (TIR files) were also provided.

CarSim 2024.1 New Features

VS Math Models

Speed Controller

The acceleration command for the speed controller, `Ax_SCcmd`, is an output variable that had no native internal calculation. A new state variable was added in this release: `SV_AX_SC`. This can be set directly (as with a parameter) or used in `EQ_` VS Commands. The output is now a combination of the state variables and the optionally imported variable. Having the state variable supports the use of an import that has the same options as other control imports (`ADD`, `REPLACE`, `MULTIPLY`, etc.).

Echo File

The Echo file generated at the end of a simulation lists all state variables and their values, providing an option to continue the run. If the model has components that include pure time delays (e.g., transport time for a hydraulic line), these components have internal buffer arrays that include saved values as needed to provide the delayed pressure multiple timestep later. If these are in effect, the End file now has comments at the bottom of the section of State Variables that list the parameters that enabled the time delays.

Real-Time

Runtime error handling

To quickly identify errors in parameter settings, system settings, and incorrect operations in RT simulations, we have improved the VS solver and VS browser. If a runtime error occurs, the VS solver will create the error log file "Error_log.txt" on the RT target machine. Users can click the button on the VS browser to view the error log file. If an error occurs during simulation, such as referencing an undefined variable or VS command, the VS solver will stop running and generate

an error log file, sending an error message back to the VS browser on the host machine. VS browser will also popup error messages.

VS Browser: Graphical User Interface (GUI)

Database Builder

The Database Builder tool was revised to identify specialty CPAR archives more easily for examples that are not included in the main groups of examples. The screen now has two options for showing CPAR archive files:

1. Show specialty archives, such as those for specific RT systems, external software, SDK, and other cases that are not applicable for most users.
2. Show all available CPAR archives, for users who wish to pick and choose specific categories of interest for a new database.

Moving Objects

The VS Browser has two screens for adding moving objects: **Multiple Moving Objects** and **Single Moving Object (Custom)**. Both have multiple options for specifying motions of new moving objects. Those options are used to write VS Commands for the new objects. In older versions, they used the commands `EQ_IN` and `EQ_OUT`. These screens were changed to use the command `EQ_STATE`, introduced in 2023.0. This is done to have all the moving objects (and possibly ADAS sensor detections) set at the start of each simulation timestep (in the `STATE` stage), eliminating a potential lag in the simulation of resulting ADAS controls (typically calculated in the following `CTL` stage).

Documentation

A new section was added to the Help menu for support of the VS SDK (Software Development Kit) with the following documents:

1. Automating Runs with the VS API
2. Extending a Model with VS Commands and the API
3. The VehicleSim API
4. Vehicle Module Simulation Integration Utility
5. VS Connect API
6. VS Connect S-Function
7. VS Output and Table API
8. vs_sf VS Connect Server
9. VS Table Tool

The following documents were added:

10. Technical Memos > Running CarSim on Linux

The following documents in the **Help** menu were updated:

11. ADAS Sensors and Moving Objects
12. Controls > Driver Controls
13. External Models and RT Systems
14. Model Extensions and RT > Model Extensions and RT Systems
15. Paths, Road Surfaces, and Scenes > VS Scene Builder
16. Tires > Terramechanics Models
17. Tires > Tire Models
18. Tools > Database Builder

The following Technical Memos were updated:

19. CarMaker Converter
20. Unreal Engine: Live Animation with Simulink

The following RT and DS documents were updated:

21. A&D Guide for VehicleSim Products
22. CANoe Guide for VehicleSim Products
23. Concurrent RT Guide for VehicleSim Products
24. ETAS RTPC Guide for VehicleSim Products
25. NI RT Guide for VehicleSim Products
26. RT-Lab Guide for VehicleSim Products
27. Speedgoat Guide for VehicleSim Products

CarSim 2024.0 New Features

VS Solver Architecture: Delayed Initialization

A new option to delay initialization was added to allow imported variables from external software such as Simulink to be used to initialize the vehicle state before the simulation starts. This is needed when variables that are critical for determining the initial state of the model are provided by user VS Commands or imported from external software. A new system parameter `OPT_DELAY_INIT` is used to choose an alternate time to perform the initialization. Details are provided in a new Tech Memo, *Initialize a Vehicle with Imported Ground Z*, and new examples were added to the database.

VS Math Models

The following options were added to the CarSim Math Model:

1. Driving in reverse is now supported when using the built-in Speed Controller in acceleration control mode (`OPT_SC` = 5).

2. An idle speed controller can be installed on IC powertrains to regulate the input throttle and automatically prevent engine stall. For more information, see the *Engine Idle Speed Controller* section in the *Powertrain Systems* Help document.
3. Siemens MF-Tyre/MF-Swift tire model is now supported on Linux 64-bit system environment.

VS Browser: Graphical User Interface (GUI)

CarMaker Converter

The CarMaker Converter is available to read CarMaker datasets and convert them to CarSim properties, setting values in a CarSim database using COM automation with the Browser. This converter has been updated with general performance improvements. If you'd like to use the converter, please contact Mechanical Simulation technical support.

Other Improvements

The menu option **File > Export to Simian** has been replaced with **File > Export ADP Package**. This action will automatically create a package of the current model that the Applied Intuition software platform (ADP) can import and use. The file extension is **.vspak**.

Real-Time Platform: Vector CANoe

The VS Solvers now support the Vector CANoe Real-Time platform, which includes VN8900, VT6000, and RT Rack. These devices together form the Vector Tool Platform and are equipped with a Windows operating system. The CANoe platform is supported by the existing VS Browser (using the **Models: Transfer to Remote RT Target** library). Example runs were added to the database and the **Real-Time and DS Systems > Vector CANoe Guide** documentation was added.

Documentation

Built-in Help Documents

The following Technical Memo was added:

1. Initialize a Vehicle with Imported Ground Z

The following Reference Manuals were updated:

2. System Parameters in VS Math Models
3. VS Browser Reference Manual
4. VS Commands Reference Manual
5. VS Math Model Manual

The following document in the **Help** menu was updated:

6. Powertrain Systems

The following Technical Memo was updated:

7. CarMaker Converter

The following RT and DS documents were updated:

8. A&D Guide
9. Concurrent RT Guide
10. dSPACE RT Guide
11. ETAS RTPC Guide
12. NI RT Target Systems
13. RT-Lab Target Systems
14. Speedgoat Guide

Japanese Language

Japanese language documentation is now available from the user section of the Mechanical Simulation website. The documents available at the time of this writing are translated from versions 2023.1 and 2023.2, with 2024.0 equivalents available later.

Database

New examples were added in the following categories.

Engine Idle Controller

A new simulation shows a vehicle on a hill with the built-in engine idle controller that prevents stalling at low speed.

Imported Ground Elevation

New simulations are added to show several methods for handling user-defined road surfaces using Simulink or VS Commands. The examples make use of the new system parameter `OPT_DELAY_INIT` that allows imported variable to be used during the initialization of the vehicle model. Details are provided in a new Tech Memo, *Initialize a Vehicle with Imported Ground Z*.

Reversing with Accel Control

A new simulation shows a vehicle with a trailer backing into a parking location using the acceleration control mode of the speed controller (`OPT_SC` = 5).

CarSim 2023.2 New Features

VS Solver Architecture: FMI 3.0

CarSim now supports the Functional Mock-up Interface version 3.0 (FMI 3.0). The VS Browser can generate a Functional Mock-up Unit (FMU/FMI3.0) as a slave in a co-simulation to run in the Windows/Linux environment. You can integrate this FMU into other simulation environments such as MATLAB/Simulink, starting with MATLAB release R2023b; dSPACE VEOS 2023a; dSPACE SCALEXIO Real-Time 2023a and others.

VS Math Models

Articulated Vehicles with One Axle in the Leading Unit

Articulated vehicles with one axle in the front part are common in agriculture and construction. They can now be represented in CarSim using as combination vehicle models in which the lead unit has just one axle. In these cases, the VS hitch is located between the first and second axles. Note that the hitch must be stiff enough in pitch to limit the relative pitch and roll of the first and second units of the vehicle.

In addition to the extended capabilities of the VS Solvers to support these vehicles, the VS Browser screen for the lead units has a new option to specify that there is no rear axle.

Examples are provided for a farm tractor and articulated dump truck.

Powertrain for Connected Vehicle Units

The powertrain models in CarSim were extended to support drive axles in two vehicle units in a combination vehicle. This extension enables the models to replicate powertrains for articulated vehicles where drive axles are in consecutive units such an articulated farm tractor. A new parameter `INCLUDE_TRAILER_PT` extends the scope of the powertrain model to include the trailer behind the unit linked to the powertrain via the `POWERTRAIN_UNIT` parameter.

Steering System Parameters

The vehicle VS Math Models support connections from the steering control to the steered road wheels using either a rack and pinion or recirculating ball steering gear. The VS Browser shows a gear ratio that did not appear in the Echo file made by the VS Math Model. The Browser was extended in 2023.2 to coordinate the information shown on the GUI screen with that shown in the Echo file, as described in the next section. The VS Math Model was also extended to include a calculated parameter `R_GEAR_PITMAN` in the Echo file to match the value set on the screen.

VS Browser: Graphical User Interface (GUI)

Articulated Vehicles with One Axle in the Leading Unit

As mentioned earlier, articulated vehicles with one axle in the front part can now be represented in CarSim math models. The VS Browser screens for lead units have been extended to specify that the leading unit does not have a rear axle. Several examples are included to demonstrate this capability.

CarMaker Converter

A tool called CarMaker Converter is available to read CarMaker datasets and convert them to CarSim properties, setting values in a CarSim database using COM automation with the Browser. The converter has been extended to include more model features, such as improved operation with incomplete CarMaker datasets, more robust behavior in general, improved tables for rear kinematics, and improved conversions for some BEV and HEV models.

Steering System

As noted above, VS Math Models in CarSim support connections from the steering control to the steered road wheels using either a rack and pinion or recirculating ball steering gear. VS Browsers have represented the initial connection with parameters based on common terminology in industry for specifying a steering gear ratio. For rack and pinion, the browser shows a C Factor; for recirculation ball, the browser shows a gear ratio defined as degrees of input per degrees of output.

In version 2023.2, the VS Browsers were extended slightly such that the ratio seen on the Browser screen also appears in the Echo file made for a simulation using the Steering dataset.

For the rack and pinion connection, the ratio is called a C Factor, and typically has different units than most nonlinear tables in the VS Math Model: mm of rack travel per revolution of the pinion gear. The steering screens in CarSim were extended in 2023.2 to support alternate units for the C Factor: mm/deg or mm/rev. If the units of mm/rev are selected, then the Parsfile written for the screen dataset includes a VS Command to change the units of the Configurable Function to use rev (revolution) for the independent variables expected by the Configurable Function.

For the recirculating ball option, the common terminology is a ratio greater than 1: input/output angle. The VS Browser has always written 1/*ratio* into the Parsfile for the dataset, where *ratio* is the value provided in the yellow field on the screen. To help verify proper transfer of content, the VS Math Models now include an extra calculated parameter `R_GEAR_PITMAN` that matches the *ratio* value shown by the VS Browser.

VS Software Development Kit (SDK)

Two new API functions were added for accessing vs_terrain data: `vs_road_get_terrain_function` and `vs_road_get_terrain_function`.

Documentation

The following Help documents were added:

1. Technical Memos > Validation of VS Models: New Vehicles
2. Technical Memos > Validation of VS Models: Regression Testing
3. Technical Memos > Validation of VS Models: The Basics

The following Reference Manuals were updated:

4. VS Commands Reference Manual
5. VS Visualizer Reference Manual

The following documents in the **Help** menu were updated:

6. CarSim and TruckSim Steering Systems
7. Paths and Road Surfaces
8. Powertrain System
9. Vehicles

The following Technical Memos were updated:

10. CarMaker Data Converter
11. Unreal Engine: Live Animation with Simulink

The following SDK document was updated:

12. The VehicleSim API

Database

New examples were added to show some of the new features applicable to articulated vehicles.

Articulated Dump Truck

New simulations are added for an articulated dump truck. The vehicle has a lead unit with one axle and a trailing unit with two axles. The vehicle steers by articulation, driven by a hydrostatic steering system. Drive torques from the powertrain are applied to the two axles on the trailing unit.

Articulated Farm Trailer

New simulations are added for an articulated farm tractor. The vehicle model replaces an example from the 2023.1 release that included a hidden second axle on the lead unit as a workaround for the VS Math Model that required at least 2 axles on the lead unit. The new version has two capabilities that were not possible before:

1. The VS Browser and Math Model support a lead unit with one axle, including automatic initialization on an uneven surface.
2. The powertrain supports 4WD with the two powered axles on different vehicle units.

CarSim 2023.1 New Features

VS Solver Architecture

Import Arrays for Connecting to External Software

VS Solvers may be run under the control of wrapper programs in simulation environments that combine external model components with the VS Math Model, exchanging information using arrays of import and export variables. The variables used to create the Import and Export arrays are defined before the simulation starts, using the `IMPORT` and `EXPORT` VS Commands. The `IMPORT` command identifies the import variable that will be added to the Import array and specifies a mode for usage within the VS Math Model.

In prior versions, three modes were available for variables associated with the Import array: `ADD`, `MULTIPLY`, and `REPLACE`. The mode could only be specified prior to the start of the simulation using the `IMPORT` command. After the run started, it could not be changed.

The mode of an Import variable may now be changed during the simulation using a new VS Command `SET_IMPORT_MODE`. In addition, a new mode `AVAILABLE` has been added. This mode indicates that the variable is in the Import array but is not currently being used by the VS Math Model. The intent is that the action of setting up the Import array has been separated from the actions of using the import variables, to allow complicated scenarios to be simulated in which

external controller or model components are enabled and disabled for different time sections within the simulation.

Timeline for Starting a Simulation

A Simulation is started by reading input files that define the layout of the model and values for parameters in the model, and then initializing the model. At a certain point, the model is "locked" in the sense that the number of ordinary differential equations (ODEs) is set. In recent versions, that point occurred at the end of the process of reading input files. When running under external control via the VS API, this step is performed by the API function `vs_setdef_and_read`.

The locking of the model now takes place slightly later, in the step performed by the API function `vs_initialize`. This allows custom wrapper programs to extend the model via API functions after input Parsfiles have been read, but before the model is initialized. For example, this change allows a wrapper to install the speed controller in support of other options used by the wrapper.

Timeline for Each Timestep

Starting with version 2023.0, the calculations done each timestep have been reorganized to perform model calculations in four specific stages each timestep:

1. **State :** the model state (position and speed state variables) is known, and information about the environment is updated.
2. **Control :** built-in driver controls are applied.
3. **Kinematics :** variables are calculated that depend on position and velocity information.
4. **Dynamics :** variables are calculated using the remaining equations in the model, including forces, moments, accelerations, and outputs that depend on these variables.

The re-ordering of internal calculations continued for the current release, such that all driver controls are now done in the Control stage, and more environmental information is available in the State stage. There are slight differences in the newer versions because lags in the controllers were eliminated.

VS Commands

Several new commands were added.

`PIC_CONTROLLER`

PID controllers can now be used in runs via the VS Command `PID_CONTROLLER()`. These controllers can be used with any symbolic math model variable to control the dynamic systems present in CarSim. The controller will calculate the error between the input variable and a setpoint, and then output a control signal according to the size of that error, as well as its integral and derivative. See the example run *Driver Model > DLC, Constant Target Speed w/ PID* and *Help > Reference Manuals > VS Commands* for more information.

`DIFFERENTIATE`

Certain built-in variables in VS Math Models can be modified on the fly by the user via `IMPORT` statements. In certain cases, it is possible to "unlink" certain variables when using the `IMPORT` …

REPLACE command to overwrite the value calculated by the VS Solver. For example, the steering wheel angle STEER_SW is calculated by integrating the calculated steering wheel angular velocity StrAV_SW. If the steering wheel angle is replaced via an import statement, the steering wheel angular velocity is still calculated internally and no longer reflects the actual steering wheel velocity used in the solver, which is the derivative of the custom variable supplied by the user.

To remedy this, a new VS math command was added in 2023.1: DIFFERENTIATE(). When used with an EQ_... command, this new command takes in a symbolic variable and numerically calculates its derivative with a backwards-looking finite difference scheme, using up to 5 time steps.

> **Note** Calculating the derivative of discontinuous signals can produce extremely large instantaneous velocities. It's recommended to use MAX() and MIN() to filter out these very large values.

SET_IMPORT_MODE

The mode of an import variables may now be changed during the simulation using a new VS Command SET_IMPORT_MODE, typically using a VS Event. A new AVAILABLE mode can be set for import variables to add variables to the array shared with external software such as Simulink, without using the variable within the math model. The mode can later be changed as needed using SET_IMPORT_MODE.

SAFE_DIV

A new function SAFE_DIV is available for user-defined expressions to avoid divide-by-zero issues.

VS Math Models

Conventional Steering System

The conventional steering system is now an optional component, installed with the math model keyword INSTALL_STEERING. When this keyword is not supplied, the steering degree of freedom of each axle, if present, is locked out, and no steering gear or column exist. As such, while driver controls will calculate a requested steering angle, this will have no effect. All the usual steering system options, including the ability to import various subsystems, remain present if the steering system is installed, with no changes in behavior or simulation results. The new option to not have the steering system defined supports vehicles which do not require a conventional, automotive steering system.

Hydrostatic Steering System

Off-highway vehicles may steer with concepts like differential steering (skid-steer) or steering by articulation (actuated hitch between vehicle halves). These are often hydraulically actuated given the forces involved. To suit these types of vehicles/steering concepts, a new steering system model is included for 2023.1, referred to as the *hydrostatic steering system*. Within this system are two available types:

- *Hydrostatic drive*, using the differential steering concept, activated by the VS Command `INSTALL_STEERING_HYDRO 0`. Each side of the vehicle has a pump-controlled rotary actuator, allowing the thrust on each side to be controlled independently. This system is also used to drive the vehicle forwards and backwards.
- *Hydrostatic articulation*, using the steering by articulation concept, activated by VS Command `INSTALL_STEERING_HYDRO 1`. The hitch articulation angle is controlled by a valve-controlled linear actuator which generates a hitch moment.

For more information on the new model, refer to Help > Steering > Hydrostatic Systems.

Terramechanics Models

VS Terramechanics: Rigid Wheel (TMRW) is now available for use on Linux and with various real-time platforms. For more information, refer to Help > Tires > Terramechanics Models.

A longitudinal contact patch dimension was added to VS Terramechanics: Rigid Wheel (TMRW). The normal contact region samples a few points along the width of the tire. Enabling a longitudinal dimension adds points along the circumference of the tire. This can help smooth the effects of rapid terrain variations. For more information, refer to Help > Tires > Terramechanics Models.

Hybrid/Electric Vehicles

The efficiency values of electric motors and generators used in hybrid and electric vehicles can now be imported and set directly via the import variables `IMP_EFF_MOTOR` and `IMP_EFF_GNRTR`, respectively.

A simulation involving an electric or hybrid vehicle can now be automatically terminated by setting the `SOC_STOP` parameter. When the vehicle's state of charge (battery level) drops below this value, the run terminates with a message written to the log file. This value should be a fraction of total battery capacity, and thus should be set between 0 and 1.

The previous versions did not provide import variables that allow an external model of the electrical system or battery. This version adds new import variables for the battery SOC (State of Charge), charging and discharging efficiencies and three other import variables which may need to import from the external model so users can implement their own battery model in an external tool (e.g. Simulink.)

The new import variables for the hybrid/electric powertrain are summarized in Table 1.

Table 1. Import variables added for the hybrid/electric powertrain.

Keyword	**Description**
`IMP_EFF_MOTOR`	Electric motor efficiency
`IMP_EFF_GNRTR`	Electric generator efficiency
`IMP_SOC_BTTRY`	Battery state of charge
`IMP_PW_BTTRY_CHRG`	Battery charge limit
`IMP_PW_BTTRY_DIS`	Battery discharge limit
`IMP_VOC_BTTRY`	Battery open-circuit voltage
`IMP_V_BTTRY`	Battery output voltage
`IMP_A_BTTRY`	Battery output current

PID Controllers

PID controllers can now be used to control various inputs to the vehicle. These controllers compute a control signal proportional to the error between two signals, the integral of that error, and the derivative of that error. See the example runs in * *PID Controllers* for more information.

VS Browser: Graphical User Interface (GUI)

CarSim Steering Screens

The **Steering** and **Steering: Virtual Steering Axis** screens have been updated to automatically set the new VS Command `INSTALL_STEERING`. This ensures that the steering system is defined prior to attempting to set any of its options or data.

Support of New Import Option

The **I/O Channels Import** screen now supports the selection of the `AVAILABLE` mode for import variables specified on the screen.

CarMaker Converter

A tool called CarMaker Converter is available to read CarMaker datasets and convert them to CarSim properties, setting values in a CarSim database using COM automation with the Browser.

VS Software Development Kit (SDK)

The VS API has been refined to support more interactions between wrapper programs and VS Math Models. In the 2023.0 and 2023.1 versions, the calculations done each timestep have been reorganized to perform model calculations in four specific stages each timestep:

1. **State :** the model state (position and speed state variables) is known, and information about the environment is updated.
2. **Control :** built-in driver controls are applied.
3. **Kinematics :** variables are calculated that depend on position and velocity information.
4. **Dynamics :** variables are calculated using the remaining equations in the model, including forces, moments, accelerations, and outputs that depend on these variables.

Support for Multiple Wrappers

The four stages of the calculations can each be associated with a separate wrapper program, with the possibility of supporting up to four wrappers that can work closely with the VS Math Model.

Support for multiple wrappers requires the following:

1. The wrappers must run in a sequence matching the stages of the VS Math Model equations.
2. The wrappers must each link to the same VS Solver library such that there is a single VS Math Model running.
3. The wrappers and VS Solver must all run on the same CPU core.

This capability is now supported for Simulink users with the inclusions of four VS S-Functions, each associated with a different stage.

VS API Changes

Import and output variables are defined in a VS Solver with API functions such as `vs_define_imp` and `vs_define_out`. These create an internal symbolic structure with an attached number (double), units, description, and other properties. New API functions `vs_define_imp_where` and `vs_define_out_where` were added for version 2023.0 to include information about where the import is used, or the output is calculated. In the current version (2023.1) all import and output variables in the VS Math Model have the stage identified. This information is obtained from machined-generated documents provided in simple text or csv spreadsheet format, viewed by screens in the VS Browser that are used to select import or output variables.

The `vs_statement` API function allows a wrapper program to apply VS Commands programmatically, rather than by reading from an input Parsfile.

As noted earlier, the locking of the VS Math Model now takes place a little bit later in the startup, such that API function calls can be made after `vs_setdef_and_read` but before `vs_initialize`. This allows more state variables to be added, including ODEs.

The VS Solver has several functions to install callback functions that can be deployed in multiple points in the simulation timeline. The main one for calculations is the calc callback, installed with the API function `vs_install_calc_functions`. There are now 12 locations each time step where callback functions can be applied, including the four stages that now exist each timestep.

Examples

A new SDK example was added to show how the speed controller can be installed programmatically. This was not possible in past versions, because the controller installed an ODE state variable needed for integral control. With the re-ordering of the simulation setup, this step can occur after the input files have been read but before the model initialization is performed.

Documentation

The following Help documents were added:

1. Steering > Hydrostatic Systems
2. Technical Memos > CarMaker Converter

The following Guides and Tutorial document was updated:

3. Running a VS Math Model in Simulink

The following Reference Manuals were updated:

4. VS Browser (GUI and Database)
5. VS COM Interface
6. VS Commands

7. VS Commands Summary
8. VS Math Models

The following documents in the **Help** menu were updated:

9. Controls > Driver Controls
10. Controls > Electronic Stability Controller
11. Controls > Trailer Backing Controller
12. Model Extensions and RT > External Models and RT Systems
13. Model Extensions and RT > Import and Export Variables
14. Run Control Screen (Home)
15. Steering > Steering Systems
16. Suspension Systems
17. Tires > Tire Models
18. Tires > Terramechanics Models
19. Tools > Running with Parallel Solvers
20. Vehicles

The following Technical Memos were updated:

21. Example: Extending a Model with VS Commands and the API
22. Example: Multiple Ports in Simulink for Sensors
23. Numerical Integration in VS Math Models
24. Unreal Engine Live Animation of a CarSim or TruckSim Solver in Simulink
25. VS Support for Multiple S-Functions
26. VS Solver CLI Wrapper
27. vs_sf VS Connect Server

The following Real-Time and DS System document was updated:

28. dSPACE RT Guide

The following SDK document was updated:

29. The VehicleSim API

Database

Hydrostatic Steering Systems

Two new example vehicles demonstrate the hydrostatic steering systems: the *skid-steer loader* makes use of the hydrostatic drive system, while the *articulated farm tractor* uses the hydrostatic articulation system. Both vehicles include examples of open and closed loop controls; the closed loop controllers are implemented using the new, built-in `PID_CONTROLLER` VS Command.

Batch Iteration

A new example uses a MATLAB script and Simulink model to iterate a run-time variable [`SPEED_TARGET_CONSTANT`], for a batch of runs.

Change Import Mode

Three new examples demonstrate Handoff steering from Manual control to the Driver Model. In the first example, the steering wheel angle is overwritten with the import variable `IMP_STEER_SW` in replace mode. An Event series is used to switch this mode to 'Add' to handoff to the Driver Model. The second example also sets the steering mode to torque steer during handoff to provide a smoother transition back to the driver model calculation. The third example sets the steering mode to torque steer and implements the new, built-in `PID_CONTROLLER` VS Command during handoff to provide a very smooth transition back to the driver model calculation.

External Battery Model

An external electrical battery model is added. The solver in this version adds new import variables for the battery SOC (State of Charge), charging and discharging efficiencies and three other import variables. The new example utilizes those new import variables and specifies the electrical system and battery in Simulink model.

PID Controllers

A new model, *Driver Model > DLC, Constant Target Speed w/ PID*, was created to demonstrate the use of the new generic PID controllers.

Parallel Solvers - Vehicle Hauling

Two new parallel solver examples demonstrate a lead unit hauling another fully modeled vehicle on a flatbed trailer. The solvers calculating each vehicle are run in parallel through Simulink, having the vehicle combination run through a DLC maneuver and a Bounce Test.

CarSim 2023.0 New Features

VS Solver Architecture

Stages of Model Calculations

The architecture of the VS Solver has traditionally had two stages available for model extensions via VS Commands and custom wrappers connecting with application program interface (API)

functions: kinematics and dynamics. Wrappers for external software tools such as Simulink dealt only with the whole model, sharing import and export variables once per time step.

The architecture was extended to organize model calculations into four specific stages each timestep:

1. **State :** the vehicle state (position and speed state variables) is known, and information about the environment is also updated (station, ADAS targets and sensors, wind, and path information for the driver model).
2. **Control :** the built-in driver controls are applied to provide steering and speed control variables.
3. **Kinematics :** variables are calculated that depend on the position and velocity information from the vehicle and controllers.
4. **Dynamics :** variables are calculated using the remaining equations in the model. ODEs for the entire VS Math Model are integrated to obtain the values available at the start of the next timestep.

The new organization was made to provide tighter connections with external software for controllers and replacements for vehicle components. It also improves options available for advanced users extending models with VS Commands and embedded Python.

Import and Export Arrays for Connecting to External Software

VS Solvers may be run under the control of wrapper programs in simulation environments that combine external model components with the VS Math Model, exchanging information using arrays of import and export variables. In past versions, communication with the simulation environment takes place once each timestep, such that the VS Math Model receives import variables before performing any calculations, and then copies output variables into an export array that is shared after all calculations in the VS Math Model have been completed.

As noted above, the calculations made in the VS Math Model each timestep are now organized into four stages, to support closer timing when extending models with external software. Features were added to support the use of multiple wrappers with a single VS Math Model:

- `OPT_MULTI_WRAPPERS` is a new user parameter available when using import/export arrays. It can be set to 0 (default) or 1. When set to 1, the VS Solver will connect with up to four wrappers connected for the simulation (one for each stage of the model calculations). Six new arrays are added (three for import and three for export) in this mode, for a total of eight arrays for exchanging information.
- `IWRAPPER` is a new hidden index parameter, managed in the same way as `IUNIT`, `IAXLE`, etc. that allows existing `IMPORT` and `EXPORT` commands to be used as needed to set exchange arrays for four wrappers.
- Each timestep, the VS Math Model makes up to four passes through the model calculations (each under the control of a different wrapper), performing only the calculations associated with the currently active wrapper.

- Machine-generated documentation files for Import variables now identify the stage in which the variable is used.
- Machine-generated documentation files for output variables now mention the stage in which the variable is calculated.

To support these new capabilities, four new Simulink S-Functions are provided in the release for Windows and non-RT Linux, as described in the VS Wrappers section (page 23).

The new process has no noticeable effect on the computation speed for the simulation.

VS Commands

Several new commands were added, and the effects of some existing commands were changed.

1. New commands `EQ_STATE` and `EQ_KIN` were added to create new equations in more stages of the model calculations. Related commands were added: `DELETE_EQS_KIN`, `DELETE_EQS_STATE`, `RESET_EQS_KIN`, and `RESET_EQS_STATE`.
2. Equations added by the `EQ_IN` command now apply after the new State stage and before the new Control stage. (Most existing examples using `EQ_IN` continue to work as originally intended.)
3. The `DEFINE_OUTPUT` command was improved to work more effectively. `SET_OUTPUT_SHORT_NAME` was added to allow the manual creation of short (ERD compatible) output names by the user.
4. The `RETURN` command used in defined functions was improved to reduce problems. `DEFINE_LOCAL` was improved to eliminate option of indexed local variables.
5. Error checking was added for lengths of new variables and parameters. The name lengths were extended to 48 characters; longer names now generate error messages.

Echo Files

The display of information in Echo files generated for each simulation has been extended to indicate which parameters or table properties have been assigned values by reading from file, as opposed to those that keep default values. Some changes in this release are:

1. The convention for identifying a parameter whose value was not specified by any of the input files is to prefix the description with `[D]` (default). This has been extended to include properties of Configurable Functions.
2. A parameter `OPT_ECHO_DEFAULT` can disable the display of the `[D]` prefix, which is sometimes helpful when monitoring differences in Echo files. Another option has been added, in which the Echo file shows only parameters and Configurable Function properties that were specified in input Parsfiles. These files are much shorter and can be convenient for some applications involving automation and large numbers of runs.

Other Improvements

1. All error messages were reviewed; many were updated to provide more specific information about the cause of the error.

2. New VS API functions were added to provide a location in the equations of the model for import and output variables; these are described in the API Reference Manual in the VS software development kit (SDK). C examples in the SDK demonstrate the new functions.

VS Math Models

Terramechanics-Based Tire Model

There is a new terramechanics-based tire model for predicting tire forces and moments when operating on soft soil. The tire is treated as rigid, in contrast to a typical CarSim or TruckSim tire (soft soil, rigid tire vs. hard pavement, flexible tire). While this type of tire model has no standard implementation, these kinds of models are often referred to as Bekker-Wong models, referring to two prominent terramechanics researchers.

The new terramechanics-based model is referred to as VS Terramechanics: Rigid Wheel (VS TMRW). The implementation makes use of the standard tire interface (VS STI) and is specific to CarSim and TruckSim, including some additional effects for use in this setting. As VS TMRW uses VS STI, there are no specific VS Browser screens associated with it. Rather, the Tires (External) and Generic Table screens are used to create and populate instances of VS TMRW.

VS TMRW is only supported for CarSim and TruckSim 2023.0 running on a 64-bit Windows operating system. For more information on other limitations of the new model and how to use VS TMRW, refer to Help > Tires > Terramechanics Models.

Trailer Aerodynamics and Crosswind

Wind has traditionally been defined using direction and speed that are calculated with Configurable Functions of time and location (station along the vehicle reference path). For vehicles with trailers, there was no easy way to account for a crosswind hitting the trailer after it reaches a station that the leading unit had already crossed. The VS Math Model now includes multiple wind Configurable Functions for simulations with multiple vehicle units. For trailers, an equivalent station variable is calculated using station for the leading unit, adjusted by the distance between the front reference point and trailing hitch point for the leading unit.

VS Tire Tester

For external tires, VS Tire Tester iteratively solves for the wheel center height needed to produce the requested tire vertical force. Beginning in version 2023.0, the maximum number of allowed iterations, as well as the solution tolerance, may be adjusted with the math model keywords `TT_MAX_ITERATIONS` and `TT_TOLERANCE_FZ`, respectively. The default values of 1000 iterations and 1.0 N are consistent with the calculations of previous versions.

Importing a Path Preview for Controllers

The new VS Command `INSTALL_PATH_IMPORT` defines 20 import variables `IMP_PATH_X`*i* and `IMP_PATH_Y`*i* (*i* = 0, 1, …, 9). These are the coordinates of 10 preview points, expressed in the vehicle coordinate system. If all 20 are activated for import using the `IMPORT` VS Command, then the steering controller and/or speed controller will not reference the built-in path (that of `PATH_ID_DM` and potentially `LTARG_ID_DM`), but instead use the imported path preview. See the Driver Controls document for further details.

Support Electric Motor on Twin-Clutch Differential

Electric motor has not been available on the twin-clutch differential axle. In this release, the solver changes to support an electric motor on the twin-clutch differential axle. Therefore, an electric motor can be installed on axle by command line or `OPT_MOTOR_ON_WHEEL 0` when `opt_twin_clutch` is 1.

Advancing the Interface to External Tire Models

In past versions, VS Math Models connected to external tire models one tire at a time to get forces and moments of each tire. Some third-party tire models support parallelized calculation of all tires. The VS Solver and external tire model interface have been extended for the parallelized calculation of the external tire model namely COSIN FTire. The new interface speeds up the simulation by about a factor of three compared with the old interface for FTire.

The external tire model interface to COSIN FTire also adopts a dynamic library loader that automatically detects the FTire solver as installed by the FTire installer.

Miscellaneous

1. In past versions, the path follower driver model was always included in the model. It has been made optional and is installed with a new command `INSTALL_DM_PATH_FOLLOWER`. As before, the DM controller can be deactivated by setting the parameter `OPT_DM` to zero.
2. The parameter `STEER_MAX_TORQUE` can be set to shut down a run if the steering torque exceeds the given value. `STEER_MAX_TORQUE` is turned off by default. A value greater than 0 will turn on the max torque limit functionality.
3. Back-spinning of wheel with electric motor is prevented when the battery is recharged with regenerative brake.

VS Wrappers

Multiple Simulink S-function Blocks

Four versions of the VS S-Function have been made that connect with the VS Math Model during different the stages of the calculations made each timestep that were described earlier (page 13): `vs_state` (State), `vs_ctl` (Control), `vs_kin` (Kinematics), and `vs_dyn` (Dynamics). The new S-Functions were added to the VS library that is accessed from Simulink. These VS S-Functions provide multiple connection points between the VS Math Model and Simulink that are applied in series each timestep by Simulink.

Any combination of two to four of the new S-Functions may be used. The timing in Simulink is such that the last S-Function is applied at the end of the timestep, and the others are applied earlier, in sequence with other blocks in the Simulink model. The result is that replacement of parts of the vehicle model (tires, springs, brake controllers, etc.) is handled inline, without the delay of waiting until the next timestep.

Examples have been added to illustrate the use of the new serial S-Functions, and a technical memo has been added to describe the operation.

dSPACE RT Unreal Live Animation and Remote Data Access

The CarSim solver wrapper for dSPACE SCALEXIO 64-bit Linux has been enhanced to utilize VS Connect to provide remote data access and synchronization capabilities.

This feature can be used to produce Live Animation of RT simulation runs using the VehicleSim Dynamics plugin for Unreal Engine. The VS Connect features of this wrapper also allow advanced users to use the VS Connect API to write custom C/C++ programs that remotely access Imports and Outputs of the CarSim solver while it is running on the RT system.

For more information, see the example Run **{RT: dSPACE} SCALEXIO: Unreal Engine Live Animation**", and the new tech memo **Unreal Engine: Live Animation with Simulink**.

Support for Speedgoat Real-Time Platform

The VS solvers now support the Speedgoat Real-Time platform: Performance and Mobile. The RT system is QNX 7.1 64bit. Software is MATLAB/Speedgoat R2021a or newer. The Speedgoat platform is supported by the existing VS Browser (using the **Models: Transfer to Remote RT Target** library). Two example runs were added to the database and the **Model Extensions and RT > External Models and RT Systems** documentation was updated.

VS Browser: Graphic User Interface (GUI)

Minor changes were made to some of the existing screens.

1. The **Models: Simulink** screen has a new drop-down control for models with three kinds of VS S-Functions:
    a. The Simulink model has a single VS S-Function for the simulation.
    b. The Simulink model has multiple VS S-Functions, each for a separate vehicle. The vehicle simulations are run in parallel in Simulink using the **Tools > Parallel VS Math Models** screen.
    c. The Simulink model has multiple VS S-Functions, each for a different stage of the VS Math Model calculations. The S-Functions are run in series to support in-line extensions to the VS Math Model from Simulink. When this mode is selected, the screen blue links to four sets of I/O ports, each for a different stage and associated S-Function.
2. The **Payload: Custom** screen has a new drop-down control to specify the reference for the Z coordinate of the payload: sprung mass coordinate system or trailer front hitch height. (This option is like one that was added in 2022.1 to the **Payload: Box Shape** screen.)
3. The TruckSim **Vehicle Dolly** screens have a similar drop-down control for locating animation shapes relative to the sprung mass coordinate system or trailer front hitch height.
4. The **Front Twin-Clutch Differential** and **Rear Twin-Clutch Differential** screen adds a drop-down control to specify either with or without electric motor on center and two electric motor links.
5. External tire interface to COSIN FTire model adopts a dynamic library loader which automatically detects FTire solver under FTire installation. Therefore, **Tire (External)**

screen has changed to remove the **Tire program file (DLL)** field when FTire model is selected.

VS Visualizer Pro

VS Visualizer Pro uses the power of Unreal Engine to bring the simulation results from CarSim to a new level of realism. A demo release is available as a separate download from www.carsim.com with a CPAR with a collection runs that can be viewed and modified in CarSim and highlight some of what is possible with this next generation of VehicleSim visualization.

VS Software Development Kit (SDK)

The `vs_output` API has been updated to allow `.mat` file I/O.

Documentation

The following documents were added to the **Help** menu:

1. Deprecated Items > Solid Axle Kinematics
2. Real-Time and DS Systems > Speedgoat Guide for VehicleSim Products
3. Tires > Terramechanics Models
4. Technical Memos > VS Support for Multiple S-Functions
5. Technical Memos > Unreal Engine: Live Animation with Simulink

The following Reference Manuals were updated:

6. System Parameters in VS Math Models
7. VS Browser (GUI and Database)
8. VS Commands
9. VS Commands Summary
10. VS Math Models

The following Screen documents were updated:

11. Animator > Vehicles and Sensor Targets
12. Controls > Driver Controls
13. Model Extensions and RT > External Models and RT Systems
14. Paths and Road Surfaces
15. Payloads
16. Setting Up Import and Output Variables
17. Suspension Systems
18. Tires > Tire Models

19. Vehicle Screens and Outputs

The following Technical Memos were updated:

20. Example: Extending a Model with VS Commands and the API
21. GPS and UTM Coordinates

The following Real-Time and DS System documents were updated:

22. dSPACE RT Guide
23. VI-Grade DriveSim & VehicleSim Dynamics

The following SDK documents were updated:

24. The VehicleSim API
25. VS Output API: Reading and Accessing VS Output Files

The following Unreal documents were updated:

26. Unreal Engine & Python Co-simulation using VS Connect
27. VehicleSim Dynamics plugin for Unreal Engine example using VS Connect

Database

Additions were made in some of the run categories. The following new categories (with associated CPAR archive files) were added.

Aerodynamics

An aerodynamics dataset has been added which uses data representative of a small van to model the aerodynamics of an enclosed rental trailer. The aerodynamic effects of the trailer are then compared with the tow vehicle alone with an updated Crosswind Test and a new Virtual Wind Tunnel test. The Virtual Wind Tunnel test calculates the aerodynamic drag, side, lift, roll, pitch, and yaw coefficients from the measured forces and moments for validation purposes.

Alternative Coordinates

VS Math Models can now generate Universal Transverse Mercator coordinates, as shown in a new example run.

Driver Model

The path preview import variables, p. 23, are demonstrated with some new examples in the *** Driver Model** category/CPAR.

Echo File Options

The Quick Start Guide example was run with different settings that affect the information shown in Echo files.

Electrified Twin-clutch Differential

An example with an electric motor on each of front and rear twin-clutch differential is included.

MF-Tyre on Simulink

Several examples extend the VS Math Model by connecting with Siemens MF-Tyre using Simulink.

Multiple S-Function Wrappers

Several Simulink models were made using the new four S-Functions for different stages of the model. Some are in the category ***Multiple S-Function Wrappers**.

There is also an example with two S-Functions in the category *** MF-Tyre on Simulink**, and another in an older category **Kinematical Preview Demo**.

New Signs

Several runs were made showing many highways signs (VS Visualizer resources) that were added to the database.

New VS Commands

Several runs were made showing new VS Command features. Multi-index variable arrays are used to simulate multiple flat tires. An improved steer controller example has been added.

Terramechanics

Several runs illustrated the new soft-soil model option based on terramechanics.

CarSim 2022.1 New Features

VS Solver: Architecture

VS Commands

1. The `PARTIAL` and `PARTIAL2` VS Commands have been added to return the partial derivative of a Configurable Function in either the row or column direction. The new `INVERSE` function returns the inverse of a Configurable Function if it is available.
2. `INSTALL_DM_IMPORTS` has been added to allow for installing Driver Model imports only. Previously, the only way to install them was to use `INSTALL_DM_OUTPUTS`, which also (still) installs them.

Other Improvements

1. The embedded Python included with all products has been updated to 3.10.2.
2. The user can now specify pre-processing and post-processing callback functions to be before and after the solver executes. These are available via the VS API or the GUI.

3. Support dSPACE SCALEXIO Linux 64-bit real-time system (dSPACE RLS 2022A and newer)
4. Support OPAL-RT RT-Lab Linux 64-bit real-time system.
5. The solver is now working with 32-bit and 64-bit LabVIEW.

VS Math Models

Powertrain Improvements

Improvements were made in the automated shifting behavior in the powertrain, which affects the powertrain behavior and the options for setting up driver control options. Most of the improvements involve the closed-loop clutch controller.

1. The location of parameters in the Echo file was adjusted to better link parameters to parts of the overall powertrain model. For example, throttle delay was moved from the Engine section (that only exists for powertrains with an internal combustion engine) to the overall powertrain system section.
2. A calculated read-only parameter `NDIFF` is shown, in support of user-defined VS Commands. This is the maximum number of differentials that might exist, given the number of drive axles created with the `INSTALL_POWERTRAIN` command.
3. Parameters `LIMIT_DOWNSHIFT` and `LIMIT_UPSHIFT` were added, to enable automatic shifting that can skip gears. For example, if `LIMIT_UPSHIFT` = 2, then the shift controller can potentially shift directly from gear 2 to 4. The default limits are 1, which replicates shifting in older versions.
4. Parameters `T_CL_START` and `T_TH_START` were added to provide more options for setting the timing for automated clutch and throttle modulation during shifting.
5. The calculations of clutch, throttle, and shift were improved to handle new requests for clutch/throttle activity if an activity was already in progress. For example, if a new shift is request before a shift in progress has ended, the new shift is started, with the start time adjusted automatically to avoid discontinuous jumps in clutch.
6. The closed-loop clutch behavior was extended to included stopping (dis-engaging the clutch at very low speed to avoid stalling the engine) and re-starting (re-engaging the clutch when attempting to accelerate). Several new state variables were added to support the new transition events.
7. A parameter `AV_ENG_LOW_CLUTCH` was added to support the automatic disengagement of a clutch at low speed if there is risk of stalling the engine.
8. A parameter `TH_MIN_CL_ACCEL` was added to support the automatic re-engagement of a clutch when accelerating from a stopped condition.
9. Upgraded some Import variables to support interactions with internally calculated values via `ADD`, `MULTIPLY`, and `REPLACE`. The Imports are: `IMP_AT_CLUTCH`,

IMP_AV_ENG, IMP_CLT_D1_2, IMP_CLUTCH_D1, IMP_GEAR_TRANS, IMP_IENG, IMP_INV_CAP_TC, IMP_MENG_REACT, IMP_MODE_TRANS, IMP_MY_OUT_D1_L, IMP_MY_OUT_D1_R, IMP_M_DIFF_D1, IMP_M_OUT_D3_F, IMP_RM_TC, IMP_R_EFF_D1, IMP_R_EFF_TR, IMP_R_GEAR_D1, and IMP_R_GEAR_TR.

Trailer Front Hitch

The origin of a trailer sprung mass coordinate system is used to locate the front hitch point, the center of gravity (C.G.) of the sprung mass, the reference locations of the axle suspension(s), and possibly the location of a rear hitch point. The X-Y-Z local coordinates of the front hitch point for a ball/pintle or generic hitch are (0, 0, H_H_FRONT) where H_H_FRONT is a parameter for the height of the point relative to the origin. Output variables have always been available for the global coordinates of the origin, as Xo_*i*, Yo_*i*, and Zo_*i*, where *i* is the number of the trailer units (2 or higher).

Additional outputs have been added in this release for the global coordinates of the front hitch point: Xo_HF_*i*, Yo_HF_*i*, and Zo_HF_*i* (where HF indicates Hitch, Front).

The location of the front hitch point can be more convenient for animation objects attached to the trailer sprung mass, such as payloads, hitch structures, and miscellaneous structures that make up the sprung mass. The Browser screens for specifying the animation assets for a trailer unit were updated to allow the new front hitch points to be used to animate items attached to the sprung mass, sometimes more conveniently than use the origin point used to locate suspension heights.

Trailer Backing Controller

The *trailer backing controller* (TBC) calculates the steering wheel angle if the vehicle is reversing, the single preview point driver model is being used, and the vehicle is a supported type. The steering wheel angle is manipulated such that the hitch articulation angle steers the trailer along the desired path.

The INSTALL_DM_TBC VS Command is used to install the TBC, typically from the miscellaneous yellow field on the closed-loop driver model screen, as this is an extension of the single preview point mode. There is one tuning parameter, the proportional control gain, with math model keyword TBC_GAIN.

For more information on the TBC, please refer to the help file for the menu item **Help > Controls > Trailer Backing Controller**.

Improvements for Moving Object Import Variables

Upgraded some Import variables for moving objects to support interactions with internally calculated values via ADD, MULTIPLY, and REPLACE. The Imports (for object #1) are: IMP_HEAD_OBJ_1, IMP_MSG_OBJ_1, IMP_PITCH_OBJ_1, IMP_ROLL_OBJ_1, IMP_S_OBJ_1, IMP_TYPE_OBJ_1, IMP_VIS_OBJ_1, IMP_V_OBJ_1, IMP_X_OBJ_1, IMP_YAW_OBJ_1, IMP_Y_OBJ_1, and IMP_Z_OBJ_1.

VS Browser: Graphic User Interface (GUI)

64-bit Version of the Browser

The browser `CarSim.exe` is a 32-bit application that runs on both 64 and 32-bit versions of Windows. As such, it can load 32-bit plug-in libraries such as the VS Solver `carsim_32.dll` but is not able to use 64-bit libraries.

Most users have been working with 64-bit versions of Windows, and many engineering software tools are now available only as 64-bit applications and libraries. For example, the last version of 32-bit MATLAB from MathWorks was 2015b. That means any recent versions of MATLAB and Simulink will work only with the 64-bit VS Solver plug-in libraries.

The 2022.1 release includes two versions of the Browser: `CarSim.exe` (still 32-bit) and `CarSim_64.exe` (64-bit). The plan from Mechanical Simulation is to drop the 32-bit versions of our tools in the 2023.0 release. (Recent releases have already included both 32-bit and 64-bit versions of the VS Solver libraries, VS Visualizer, and other tools.)

Mechanical Simulation recommends using the 64-bit version unless there is a need to maintain compatibility with 32-bit tools. Given that recent versions of MATLAB/Simulink are only 64-bit, there is slightly better compatibility if the runs made without Simulink use the same VS Solver library as the runs made with Simulink.

Improvements in Existing Library Screens

Existing Library screens were modified.

Sprung Mass screens

VS Math Models have not included dimensions of the sprung masses, as they are not part of the equations of motion. However, in version 2020.1, VS Math Models could attach target objects to sprung masses so they can be detected by ADAS sensors. Dimensions were added to the VS Math Models (they are `LEN_SM`, `WID_SM`, and `HT_SM`) and those dimensions are available to define sizes of moving objects or for advanced users to include in VS Command equations.

The sprung mass screens have always had three yellow fields for dimensions used by VS Visualizer to scale animation assets (these dimensions are `X_LENGTH`, `Y_LENGTH`, and `Z_LENGTH`). In versions 2020.1 to 2022.0, the values from these yellow field were written twice, using both sets of keywords.

In 2022.1, the three library screens for sprung masses were extended to explicitly provide two values for each dimension: one set is for scaling animation assets for the video and the other set is for sizing a target moving object if one is attached to the sprung mass. The library screens are:

1. Vehicle: Sprung Mass,
2. Vehicle: Sprung Mass (from Whole Vehicle), and
3. Vehicle: Trailer Sprung Mass.

I/O Channels: Write

The Browser now supports access to pre-processing and post-processing callback functions (I/O Write screen) so the user has the option to execute their own programs before or after the solver is run. As mentioned above, this capability can also be accessed with the VS API.

The filename suffix option has been restored for auxiliary outputs.

Animation: Vehicles and Targets

As mentioned earlier, the VS Math Model now provides X-Y-Z coordinates for a front hitch point for trailers. This screen has a drop-down control to choose whether animation assets should be located relative to the sprung mass origin point (typically close to the ground) or at the front hitch point. The setting is written to the Parsfile for the screen with the keyword `OPT_TRAILER_REF_FRAME`. This is used when a run or video is made from the **Run Control** screen: the value of the parameter used by the Browser to generate Reference Frames for VS Visualizer using the selected point. Note that the keyword is only used within the Browser; it is not recognized by the VS Math Model.

Suspension: Compliance (Nonlinear)

The **Suspension: Compliance (Nonlinear)** screen has been updated to include a checkbox for the dynamic X compliance option for independent suspensions. When checked, the Parsfile includes `DEFINE_SUSP_X_DOF` to activate the option. Additionally, the yellow field for quasi-static longitudinal compliance is replaced with a yellow field for the dynamic longitudinal stiffness.

Miscellaneous Changes

1. Changes were made in Powertrain screens such that unused options are not installed. For example, the `INSTALL_ENGINE` command is not applied for electric powertrains.

2. The **Control: Clutch Shifting Timelines (Closed Loop)** screen was modified to include yellow fields for the new parameters `T_CL_START` and `T_TH_START`.

3. The **Powertrain: Engine** screen includes a new yellow field for a minimal speed setting for the clutch controller, with keyword `AV_ENG_LOW_CLUTCH`. If the powertrain includes a clutch operating with the built-in closed-loop controller, this engine speed might be used to dis-engage the clutch at very low speed to avoid stalling the engine.

4. The **Powertrain: Electric Motor Torque** screen includes a new checkbox and a new yellow field for an optional reduction gear. If the checkbox is checked (default is unchecked), the yellow filed appears to set the reduction gear ratio which is used to scale the input and output of the motor torque configurable table, i.e. `SPIN_SCALE_M_MOTOR_MAX` and `MMOTOR_MAX_GAIN`. Also, the gear ratio is used to modify the motor rotor inertia (`I_MOTOR`).

5. The **Control: Shifting (Open Loop)** and **Control: Shifting (Closed Loop)** screens were both changed to allow only three kinds of Configurable Functions that are appropriate for gear as a function of time: Constant, Table (steps, flat-line extrapolation), and Equation.

6. Two more plot links were added to the **Generic VS Commands** screen.

7. The **Tools** menu was modified to clarify the searching of existing runs for uses of the dataset currently in view.
8. The **Control: Steering by the Closed-Loop Driver Model** screen automatically applies the VS Command `INSTALL_DM_IMPORTS` if the single-point preview is selected (`OPT_DM = 3`). The Imports are not activated, but they are available. Previously, this did not occur and there was no easy way to access the imports.

VS Visualizer

VS Visualizer has added a preferences option to force X or Y plot axis labels to show. Users with a small VS Visualizer window and many plots may have hidden axis labels due to automatic plot window scaling. Forcing the axis labels to show will allow users to view VS Visualizer at their preferred window size.

Licensing

The Command-Line License Manager can now run as a Windows Service, allowing for the application to be started automatically when the system is booted. Additionally, running the License Manager as a Service allows for the application to be started, paused, or stopped using the Microsoft Management Console.

Simulink S-function Wrapper VS Connect Server

The Simulink s-function wrapper `vs_sf` has been enhanced with the addition of an embedded VS Connect server. Parsfile keywords added to yellow fields in the VS Browser can be used to enable and configure the VS Connect server within the `vs_sf` s-function. When enabled and configured via Parsfile parameters, this feature provides access for remote VS Connect Nodes to set imports and retrieve outputs of the VS Solver as it executes within Simulink.

This feature can also provide remote VS Connect read/write access to other data (signals) within Simulink that are external to the VS Solver by employing custom solver imports/exports for this purpose.

Examples of remote VS Connect clients that can connect to `vs_sf` include:

- The VS Connect Client S-function (`vs_connect_sf`), which may be running in the same Simulink model, or in a separate model which may itself be running on a separate computer.
- Custom C/C++ code utilizing the VS Connect library (available in the VS SDK).
- Custom Python code utilizing the VS Connect Python wrapper (available in the VS SDK).
- The forthcoming release of the VehicleSim Dynamics Plugin for Unreal Engine, which will include components that can be easily configured to provide “Live Animation” within Unreal Engine of VehicleSim solvers running remotely within Simulink.

For an example of how to configure the VS Connect server of `vs_sf`, see the example Run:

```
{External Control, Wrappers} Unreal Engine Live Animation
```

Documentation

The following documents were added to the **Help** menu:

1. Controls > Trailer Backing Controller
2. Technical Memos > Change Units of VS Math Model Variables
3. Technical Memos > vs_sf VS Connect Server
4. Tools > Database Builder

The following Guides and Tutorial document was updated:

5. Quick Start Guide

The following Reference Manuals were updated:

6. System Parameters in VS Math Models
7. VS Browser (GUI and Database)
8. VS Commands
9. VS Commands Summary
10. VS COM Interface
11. VS Math Models
12. VS SDK: The VehicleSim Software Development Kit
13. VS Table Tool
14. VS Visualizer

The following Screen documents were updated:

15. ADAS Sensors and Target Objects
16. Aerodynamics
17. Animator > Camera Setup
18. Animator > Shapes and Groups
19. Animator > Reference Frames
20. Animator > Sounds
21. Animator > Vehicles and Sensor Targets
22. Brake System
23. Controls > Driver Controls
24. Controls > Electronic Stability Control (ESC)
25. Generic Data > Generic Data Screens
26. Generic Data > Generic Table

27. Generic Data > External Parsfile
28. Hitch
29. Model Extensions and RT > Custom Forces and Motion Sensors
30. Model Extensions and RT > External Models and RT Systems
31. Model Extensions and RT > Import and Export Variables
32. Model Extensions and RT > Path Detectors
33. Paths, Road Surfaces, and Scenes > Paths and Road Surfaces
34. Paths, Road Surfaces, and Scenes > Road Surface Visualization
35. Paths, Road Surfaces, and Scenes > VS Terrain
36. Payloads
37. Plot Setup
38. Powertrain > Electric and Hybrid Electric Systems (BEV/HEV)
39. Powertrain > Engine Mounts
40. Powertrain > Powertrain System
41. Procedures and Events
42. Steering Systems
43. Suspension Systems
44. Tire Models
45. Tools > Atlas GPS Tools
46. Tools > Calculator Screen
47. Tools > Calculator Tool for Tables
48. Tools > VS / ERD File Utility
49. Vehicles

The following Deprecation Memo was updated:

50. Powertrain External Transmission Screens

The following Technical Memos were updated:

51. HPC Licensing
52. Numerical Integration in VS Math Models
53. Validation of VS Vehicle Models
54. VehicleSim License Manager (VSLM)
55. VS Solver Wrapper

The following Real-Time and DS System documents were updated:

56. RT-Lab Guide
57. VI-Grade DriveSim & VehicleSim Dynamics

The following SDK documents have been updated:

58. The VehicleSim API
59. The VS Vehicle Module Simulation Integration Utility
60. VS Output API: Reading and Accessing VS Output Files
61. VS SDK: The VehicleSim Software Development Kit

The following miscellaneous document was updated:

62. VehicleSim Dynamics plugin for Unreal Engine example using VS Connect

Database

Additions were made in some of the run categories. The following new categories (with associated CPAR archive files) were added.

Articulated bus

Examples provided with the previous release (2022.0) were re-done to better show the articulations of the connection.

Automatic clutch control

Improvements in the built-in closed-loop controller for mechanical clutches are shown for some examples involving bringing the vehicle to a start and then restarting.

Dynamic X compliance examples

The two examples which use the dynamic X compliance for independent suspensions have been updated to use the new checkbox on the **Suspension: Compliance (Nonlinear)** screen (p. 32). These examples are:

- Suspension and Ride Tests > Extended Tires, X DOF: Small Bump
- Suspension and Ride Tests > MF Tyre, X DOF: Small Bump.

Impaired driver

The closed-loop driver model includes a parameter which can be used to delay the application of the calculated steering input by the desired time. New examples have been added which make use of this parameter to model the effects of an impaired driver. The preview time is also shortened proportionally to include an impaired driver's reduced ability to focus on the road ahead.

Parametric sweep

An example uses the new pre- and post-processing capability added to VS Math Models and supported by the I/**O Channels: Write** screen.

Trailer backing controller

Various trailer backing examples are now updated to use the built-in trailer backing controller (TBC); these have been re-organized into the *** Trailer Backing Controller** category/CPAR. For more information on the TBC, please refer to the Trailer Backing Controller help file, located in Help > Driver Controls.

VS Command Examples

Copies of the Quick Start run were made to add outputs generated using new VS Commands Inverse (inverse of a Configurable Function) and Partial (partial derivative of a Configurable Function expression).

CarSim 2022.0 New Features

VS Math Models

Powertrain Improvements

Series and Parallel Hybrid Models

The powertrain model adds two types of hybrid electric models to an existing hybrid model: one is *Series Hybrid* (also known as "REEV: *Range-Extended EV*) and the other is *Parallel Hybrid*. The series hybrid system involves an engine directly connected with a generator which charges the electric battery whereas separate motor(s) drive the wheels. On the other hand, in the parallel hybrid system, an engine and motor(s) are parallel structure that the engine and motor(s) drive the wheels through the transmission when a clutch is engaged while the wheels are driven by only motor(s) when the clutch is disconnected.

The existing hybrid model which involves a planetary gear is renamed as *Power-Split Hybrid* (`OPT_HEV = 1`) as distinguished with the new series hybrid (`OPT_HEV = 3`) and parallel hybrid (`OPT_HEV = 4`).

Powering trailer axles

Articulated busses and some other combination vehicles are powered by driving trailer axles, rather than axles on the lead unit. When a powertrain is defined in a combination vehicle, a new parameter `POWERTRAIN_UNIT` specified which unit contains the powertrain. The default value is 1 (the lead unit), and the parameter is hidden from the Echo file and ignored if the vehicle does not include a trailer.

Trailers with Moving Parts

New options were added to support two kinds of connections for trailers that involve moving parts. One is for a ball or pintle hitch connected to a trailer with a hinged tow bar. These are used for truck dollies to avoid hitch loads on the pintle. Another is for articulated busses, which use an articulation system with a hinge connecting to the lead unit from a structure that is attached to the trailing unit with an articulation joint.

In support of these new connection options, the calculations made for hitch connections were redone. A new command OPT_HITCH_TYPE sets the type of hitch to 1 (generic or fifth wheel), 2 (ball or pintle hitch), 3 (ball/pintle connected to a massless tow bar, or 4 (articulation system). The first option (OPT_HITCH_TYPE = 1) provides the same model and associated outputs that were available in past versions.

Generic and fifth-wheel hitches (OPT_HITCH_TYPE = 1)

The generic hitch calculates rotations that occur on a fifth wheel in which a pitch hinge (Y rotation) is attached to the leading unit, an articulation hitch (Z rotation) is attached to the trailing unit, and the intermediate roll direction (X rotation) is defined by the vector cross product Z x Y. These angles, based on the rotation sequence Y-X-Z (also called 2-1-3) are not the same ones defined by ISO and SAE for defining sprung mass orientations; those start with yaw (Z rotation), pitch (Y rotation), and finally, roll (X rotation).

The generic hitch model calculates the 2-1-3 hitch angles based on differences in the sprung mass 3-2-1angles, then calculates moments based on those angles and their rates, applies the moments to the leading and trailing bodies using the hitch 2-1-3 axes.

Ball and pintle hitches (OPT_HITCH_TYPE = 2)

Ball and pintle hitches apply forces to connect a point in the rear of the leading unit with a point in the front the trailing unit. No moments are calculated from the angles. When the new type 2 hitch is specified, the moment calculations are skipped. Also, parameters, tables, and output variables associated with those calculations are not added to the VS Math Model.

Hinged tow bars (OPT_HITCH_TYPE = 3)

A new option (type 3) was added to support the simulation of trailers with hinged tow bars. This option adds a massless tow bar that is hinged in pitch, such that the vertical force (perpendicular to the bar) is zero. With this option, the VS Math Model calculates a relative pitch angle of the bar such that the normal (vertical) force is zero. The hitch still applies forces laterally and in the longitudinal axis of the hinged bar.

An additional parameter is defined with this option (LX_TOW_BAR) and outputs are generated as needed to animate the moving tow bar.

Articulation systems (OPT_HITCH_TYPE = 4)

Another new option (type 4) was added to support the simulation of articulated busses, where the hitch geometry is set to reduce the vertical space needed to separate the pitch and articulation joints. The distance is horizontal (rather than vertical as with most heavy-truck fifth wheels). This distance might be significant.

This option adds a massless structure to the trailer that articulates and connects to the leading unit with a hinge that allows relative pitch. The origin of the sprung mass coordinate system is moved in the X-Y plane of the trailer sprung mass such that the articulation point remains behind the hitch point the leading unit by the distance LX_ART (a new parameter for this type of hitch).

Along with the new parameter, new outputs are generated as needed for this type of hitch. All the 2-1-3 moments calculation for the generic (type = 1) hitch are also applied with this type.

Wheel Center and 3-1-2 Definitions for the Independent Suspension

There is a new option for the generic/independent suspension model, keyword `OPT_IND_KIN`, available from a checkbox on the **Suspension: Independent System Kinematics** library screen.

When `OPT_IND_KIN` is on (=1), the longitudinal movement, lateral movement, dive, and camber kinematics tables are interpreted with definitions more closely matching physical measurements or general simulation outputs. Specifically, the wheel carrier is oriented relative to the sprung mass by a 3-1-2, steer-inclination-dive rotation sequence, where the steer and inclination are given by the toe and camber input tables. The wheel carrier is then translated by the translational input data, meaning that data gives the wheel center displacement. The option works by generating another set of kinematics tables which allow the model to match the position and orientation given by the input tables' alternate definitions. This capability is enabled by the built-in Independent Suspension Kinematics Utility (IKU), which handles the conversion calculations.

`OPT_IND_KIN` off (=0, the default) retains the previous behavior exactly. The effects of the option are readily compared with four new examples in the "* Independent Suspension Kinematics" category:

- Kinematics: Bounce (`OPT_IND_KIN=0`)
- Kinematics: Bounce (`OPT_IND_KIN=1`)
- Kinematics: Roll (`OPT_IND_KIN=0`)
- Kinematics: Roll (`OPT_IND_KIN=1`).

For more information, refer to the updated help file, **Help > Suspension Systems**, especially the new "Suspension Kinematics in the Math Models" section.

GPS Calculations

GPS coordinates are calculated and provided as output variables for the first vehicle unit and for moving objects. The conversion from global X and Y coordinates in the simulation model coordinate system to GPS have been based on the starting location of the vehicle, with updates occurring when there is a significant change in GPS latitude. Several improvements were made to accommodate simulations involving multiple vehicles and moving objects that might be separated by significant distances.

1. Simulations involving more than one vehicle running within a single VS Solver now include GPS outputs for all vehicles, rather than just the first vehicle.
2. The GPS conversion parameters `GPS_REF_LAT`, `GPS_REF_LONG`, `GPS_REF_X`, and `GPS_REF_Y` now retain their initial value, reflecting the value associated with the creation of the road or scene. Instead, when any reference point is reset for a vehicle or moving object, the run's log file will contain a line indicating the vehicle or moving object ID and the latitude/longitude at the reset.

3. The GPS reference point is now checked at initialization, to improve GPS output accuracy in cases where the initial location of the vehicle or moving object is past GPS_RANGE_Y.

4. Moving object GPS output calculations now account for the reference point reset implied by the parameter GPS_RANGE_Y, rather than always using the reference point established by GPS_REF_LAT, GPS_REF_LONG, GPS_REF_X, and GPS_REF_Y.

5. Moving object GPS output calculations now function if the X-Y coordinates of the moving object are specified directly. Previously, moving object GPS outputs were only produced if the moving object was set to follow a path.

Miscellaneous Features

1. A new read-only parameter DUALS was added to the CarSim Solver to indicate whether the model supports dual tires. If duals are supported, DUALS has a value of one; otherwise, it is zero. This was done to support use of the IF statement in VS Commands for advanced applications.

2. A new output variable RRsurf_*t* was added for each tire *t* (e.g., L2i for the left inner tire on axle 2, if there are duals). This is the rolling resistance coefficient due to the ground/road surface, which can vary with location.

3. A new system parameter OPT_ECHO_DEFAULT is available to disable the identification with the indicator [D] in the Echo file for default values that were not set by reading an input file. This can be helpful in advanced applications, such as when a new run is made using an Echo file written at the end of a previous run. In this case, the [D] indicator never appears in the Echo files for the continuation run. The new parameter may be used to ensure the [D] indicator doesn't appear in other runs either, simplifying the use of text editors to compare files.

4. New import variables to control steering systems were added for each axle. These are IMP_DSTEER_CON_(*axle*) and IMP_STEER_CON_(*axle*) for recirculating ball-type systems and IMP_DSTEER_RACK_CON_(*axle*) and IMP_STEER_RACK_CON_(*axle*) for rack and pinion systems. These variables define the position and speed of the steering gear output, which is provided as input to the non-linear kinematics for the wheels. Previously, the inputs to the wheel kinematic tables could be imported with separate variables for each wheel. Since separate variables for each wheel did not imply a single position for the steering output, information about the input gear could not be inferred. With a single import variable for each, the input information can be inferred. The separate variables are still supported. You should use either the single variables for each axle or the individual variables for each wheel, but not both, depending on your needs.

5. A new output variable for dive angle was added for each wheel, DiveG_<wheel>. This is a general measurement of dive of the steered wheel hub (aka wheel carrier) which is available for all suspension types. The definition is the wheel spin change due to the suspension kinematics and compliance, measured with the outboard brakes locked. This definition matches that used by the (CarSim-only) virtual steering axis suspension model but is different from the dive definitions used internally by the independent or solid axle suspension models. (The pre-existing output variable Dive_<wheel or axle> uses the dive

definition matching the suspension type for that wheel or axle. See also the Backwards Compatibility document for a bug fix related to the dive output for solid axles.)

6. An extension to the driver controls is now supported for advanced users developing autonomous driving controls and ADAS. Activated by setting `OPT_DM_AUX` to 1. When activated, this feature produces output variables for a "bicycle model" steer angle to steer toward a path target offset (`LTARG`) different from the one used by the internal driver path follower model. See the Driver Controls help document for details on its operation and use.

VS Browser: Graphic User Interface (GUI)

Tool: Find links to the current dataset from any library

A new command was added to the **Tools** menu for searching: **Find All References to this Dataset in "Run_all.par" Files by Library...**

The existing **Find All References to This Dataset** option finds datasets that are immediately referenced by others. The search can be repeated, to find datasets that reference datasets that reverence the current dataset, but this form of manual repeating is time consuming.

The new search option takes advantage of `Run_all.par` files that are made automatically whenever a run is made, or VS Visualizer is used to view results. This has significant advantages:

1. Searching these files is rapid compared to searching the entire database. Even with large databases, the search is much less than a minute.
2. The searching finds all references where the current dataset was used in an existing run.
3. Results are filtered to show only cases where the current dataset referenced (no matter how indirectly) by datasets in a specified library.

For example, perform a search from a **Tire** dataset to find all the **Run Control** datasets that may somehow make use of the current tire dataset. Or, search for all vehicle datasets that used the current tire dataset in a run.

Support of New Joint Types

The new Hitch and Joint options are supported by changes to several screens.

1. The two Hitch screens (**Hitch: Joint Assembly** and **Hitch: Parameters**) both include a new checkbox to indicate that the hitch will generate torsional resistance, as needed for hitches of type 1 (generic or fifth wheel) and 4 (articulation system). If unchecked, the hitch will be either type 2 (ball or pintle) or 3 (pintle with hinged tow bar).
2. The **Vehicle: Trailer Sprung Mass** screen has a drop-down control to specify an additional moving part. The options are:
3. Trailer hitch joint is at a single point (default)
4. Trailer has a hinged tow bar
5. Trailer has an articulation system

The type of hitch is set by combining the data from the two types of screens. The Hitch screen (whose Parsfile is read first by the VS Math Model) set the hitch type to 0 or 1. The Sprung Mass screen then leaves the hitch type intact (option a), converts a ball/pintle hinge to type 3 (option b), or converts a generic hitch to an articulation system (option c).

Powertrain: Hybrid/Electric System screen

Powertrain: Hybrid/Electric System screen is changed to support three types of hybrid electric (Power-split, Series and Parallel hybrid) and pure electric systems. As VehicleSim used to support only one type of hybrid (power-split) and pure electric in the previous versions, this screen used to distinguish them by a checkbox (Show EV parameters only.) The new screen replaces the old checkbox with a new drop-down menu which involves four types of hybrid/electric system. The checkbox status in the data from the previous versions is automatically assigned to the new drop-down menu item.

The hybrid/electric system used to be installed by Powertrain screens (e.g. **Powertrain: Rear-Wheel Drive** screen, etc.) using keyword, `OPT_HEV`. From this version, Powertrain screens only define `OPT_HEV = 0` when IC (Internal-Combustion) powertrain is selected while **Powertrain: Hybrid/Electric System** screen defines `OPT_HEV = 1` through `4`.

Miscellaneous Changes

- Users can now specify the results output directory of an FMU with self-contained mode enabled.
- Various vehicle screens with Miscellaneous blue links can now link to **Animator: Reference Frame** datasets (in support of adding moving parts to the animated vehicle).
- Many minor changes were made to fix bugs, correct typos, improved consistency, etc.

VS Visualizer

Added new option to scale plots non-symmetrically by pressing the "alt" key while dragging with the middle-mouse-button (or left+right mouse buttons). In this mode, moving the mouse up/down zooms vertically, moving left/right zooms horizontally. There is also an option on the plot window right-click context menu ,"Asymmetric mouse zoom", to switch which mode is default (active w/o the Alt key).

VS Solver Wrapper (Command Line Interface)

Support of VS Solver Wrapper with a command-line interface has been improved to provide more options when using external software to provide automation.

- VS Solver Wrapper can now execute simulations using Simulink completely using the command line interface. Simulink will be automatically launched when the `SIMULINK_MODEL_FILE` keyword is detected in the "`all.par`" input file. For more information, see VS_SolverWrapper.pdf.
- VS Solver Wrapper can execute simulations of an FMU using the command line interface. An FMU must be FMI version 2.0 and self-contained that includes all input files and binary solvers: vehicle solver 32/64 bit for Windows and Linux 64 if running

on Linux 64, `simfile.sim`, `Run_all.par`, events parsfiles, vs_terrain file, external tire files, see VS_SolverWrapper.pdf.

- VS Solver Wrapper can now load and run External Tire model datasets. Previously these were referenced through absolute path, due to third-party requirements. They are now referenced through relative path, with the absolute path being determined dynamically through the PROGDIR simfile parameter.

Live Animation Solver Wrapper

CarSim and TruckSim have a new wrapper program, named wrapper_live_animation, which uses VS Connect to provide live animation data to VS Visualizer while the simulation run progresses. An example dataset that demonstrates this wrapper is included with CarSim and TruckSim.

This wrapper accepts command line parameters to control the run and VS Connect network connection. For a list of available command line options, use the Windows' Command Prompt to execute the following:

```
wrapper_live_animation_Release_Win32.exe -?
```

Both 32- and 64-bit versions are provided.

The wrapper also accepts keyboard input during the simulation to control the simulation (speed up, slow down, pause), as well as other options. See on-screen help when the wrapper is run for more details.

Source code (C++) and Visual Studio 2015 project files are included with this wrapper so that it can be modified or used as the basis for integrating CarSim and TruckSim solvers into external simulation programs with support for VS Connect data synchronization and live animation with VS Visualizer.

This wrapper can be found in the `Extensions\Custom_C` folder of your CarSim or TruckSim database. It is also included in the VS SDK in the folder `Libraries\vs_connect\example`.

Documentation

The following document was added to the **Help** menu:

1. High Performance Computing (HPC) and VehicleSim

The following Reference Manuals were updated:

2. VS Browser (GUI and Database)
3. VS COM Interface
4. VS Commands Reference Manual
5. VS Commands Summary
6. VS Math Models Reference Manual
7. VS Visualizer Reference Manual

The following Screen documents were updated:

8. ADAS Sensors and Target Objects
9. Animator: Wheel Arrows and Other Indicators
10. CarSim and TruckSim Brake System
11. CarSim and TruckSim Suspensions
12. CarSim Steering Systems
13. Driver Controls
14. External Models and RT Systems
15. Powertrain System
16. Powertrain for Electric and Hybrid Electric Vehicles (BEV/HEV)
17. Trailer Hitches
18. Vehicle Screens and Outputs

The following Technical Memos were updated:

19. High Performance Computing (HPC) and VehicleSim
20. HPC Licensing
21. VS Camera Sensor Simulink Block
22. VS Solver Wrapper

The following Real-Time and DS System documents were updated:

23. dSPACE RT Guide
24. Windows DS for CarSim and TruckSim

Database

New and Updated Examples

New trailer backing example

There is a new trailer backing example, representing an alley dock maneuver for a midsize pickup with one-axle rental trailer. This uses the Path Detector and VS Commands to implement a custom steering controller. The example is in the category of **Driver Model** and is named **Alley Dock (Path Detector)**.

Kept one significant figure in steering system inertia

Round the steering System Inertias to one significant figure which serve as a good estimate to this property.

Series and Parallel Hybrid Electric examples

Several new hybrid electric powertrain examples are added under each of *** Parallel-Hybrid Electric** and *** Range-Extended Electric (Series Hybrid)** categories.

Articulation system joint and power on trailing unit

Articulated bus examples are used to demonstrate two new features in the modeling available in CarSim:

1. Powertrains may be installed on a trailer in a combination vehicle (as an alternative to the lead unit). An articulated bus example powers the axle on the trailing unit.
2. Some articulated busses use articulation systems in which a pitch hinge that connects to a leading unit is located some distance in front of an articulation hinge.

ACC Perception Logic Update

The adaptive cruise control (ACC) perception logic in the ego vehicle used within the ACC Euro NCAP examples and ADAS and Active Safety examples has been updated. This perception update determines if the sensed object ahead of the ego vehicle is in the lane of the ego, or not; this is especially useful on curved roads to help reduce false-positive detections.

The four-lane (*ACC, 4-Ln Road, Fwd and Opp. Traffic*) and five-lane (ACC, 5-Lane Road, 3 Lanes Fwd Traffic) ADAS and Active Safety examples with ACC have included the same ACC perception logic updates, as included in the ACC Euro NCAP examples. The four-lane example now includes a heads-up-display (HUD) indicator in the VS Visualizer for when a vehicle has been detected in the same lane as the ego vehicle. This same logic is included in the five-lane example, though the in-lane detection indicator is not shown in the VS Visualizer.

ACC Euro NCAP Updates

The ego vehicle, global vehicle target (GVT), and other moving object setups have been updated, using VS Commands. Lastly, the Car-to-Car Stationary, Curved Road ACC Euro NCAP test has been added to the other ACC Euro NCAP examples.

ADAS: Hwy. w/Traff.,Strong LKAS and Weak LKAS Example Updates

The *Comb. ADAS: Hwy. w/Traff.,Strong LKAS* and *Comb. ADAS: Hwy. w/Traff.,Weak LKAS* examples also include the updated ACC logic with AEB turned on, as well. The Procedures for these examples have been updated with newer traffic acceleration and deceleration profiles, as well as new Events that promote the ego vehicle to continue driving on the highway, after an AEB stop.

The autonomous emergency braking (AEB) Euro NCAP tests have been updated to include the latest AEB logic, featured in CarSim, as well as the Global Vehicle Target (GVT).

Transient Response Time Examples

Three *Transient Response Time* examples, one each in the categories of *Handling and Stability Tests*, *SAE: FSAE Examples*, and *SAE: Baja*, were updated to address some VS Commands typos that were introduced in version 2021.1. The typos made the VS Commands definitions of the outputs `rt_ay`, `rt_roll`, and `rt_avz` incorrect.

New Animator Resources

New Animator Shape Assemblies have been added for the three-Axle Sleeper Cab, Flatbed Trailer, Shipping Container Hauler, and 53-ft. Box Trailer.

CarSim 2021.1 New Features

Database Delivery

In past versions of CarSim, the installer application provides an example database with hundreds of examples covering basic and advanced vehicle dynamics, ADAS, VS Visualizer animation scenarios, basic and advanced uses of built-in controllers, examples using Simulink, LabVIEW, and FMI, and other examples that might be of interest to many users. For example, the 2021.0 release included a little over 400 example runs.

Examples were available separately for specialty setups, such as real-time (RT) systems involving target machines, software development kit (SDK), desktop driving simulator (DS), examples for SAE student competitions, and others.

The delivery method has been changed for 2021.1. As with all versions in the past 15 years, the installer program installs a folder `CarSim_Prog` with our complete program stack (`carsim.exe` GUI Browser, VS Solver and other support libraries, VS Visualizer, VS Scene Builder, other utilities) and static resources such as animator 3D assets and Help PDF files. However, the installer no longer installs a default database. Instead, about 650 examples are provided in over 60 Consolidated Parsfile (CPAR) archive files. Once CarSim is installed, end users may build databases at any time, using any combination of the CPAR archives.

7. The new tools for installing databases are described in the following section. More information about the specific datasets is provided in the last section (The VehicleSim API
8. VS Output API: Reading and Accessing VS Output Files

The following Unreal documents were updated:

9. Unreal Engine & Python Co-simulation using VS Connect
10. VehicleSim Dynamics plugin for Unreal Engine example using VS Connect

Database, page 27).

VS Browser: Graphic User Interface (GUI)

The new options for building and managing databases are handled by the VS Browser `carsim.exe`, using a few new features.

Recent Database Window

When opening recent versions of CarSim, a window appears listing database folders that are known for the current version. This window (Figure 1) has been extended for 2021.1, providing access to a new tool: the Database Builder.

When launching CarSim 2021.1 for the first time, the **Select Recent Database** window shows a message describing options for building a new database or accessing older databases for use in the new version (1). If you have CarSim databases from previous versions, you can check the box **Include older versions in the list** (2) and the window will show databases that were used in the previous version of CarSim. As in past versions, any of those databases may be selected for use in the new version.

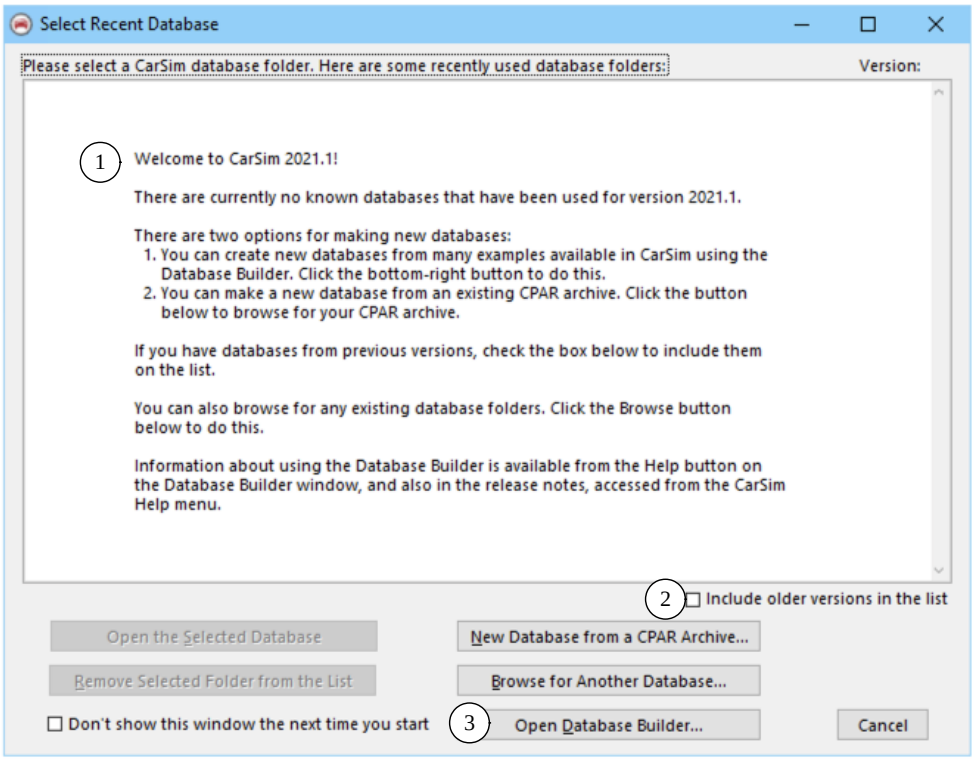

Figure 1. Recent Database window shown when opening a new version of CarSim.

Database Builder

To access examples for 2021.1 when you start CarSim, click the button **Open Database Builder** (3) (Figure 1) to bring up the Database Builder window (Figure 2). When a group of data is selected (1), the button **Build Database from Selected Items** (2) becomes active. The **Help** button (3) opens a PDF document that describes the options.

> **Note** The Help document is also available from the CarSim **Help** menu in the **Release Notes** submenu.

You can access the Database Builder any time after the initial installation of CarSim. It can be viewed whenever the **Select Recent Database** window appears. The Database Builder can also be brought up directly from within CarSim using the **File** menu (1) command **Open Database Builder** (2) (Figure 3).

Note Three of the first four options in the **File** menu correspond directly to three of the buttons in the **Select Recent Database** window (Figure 1).

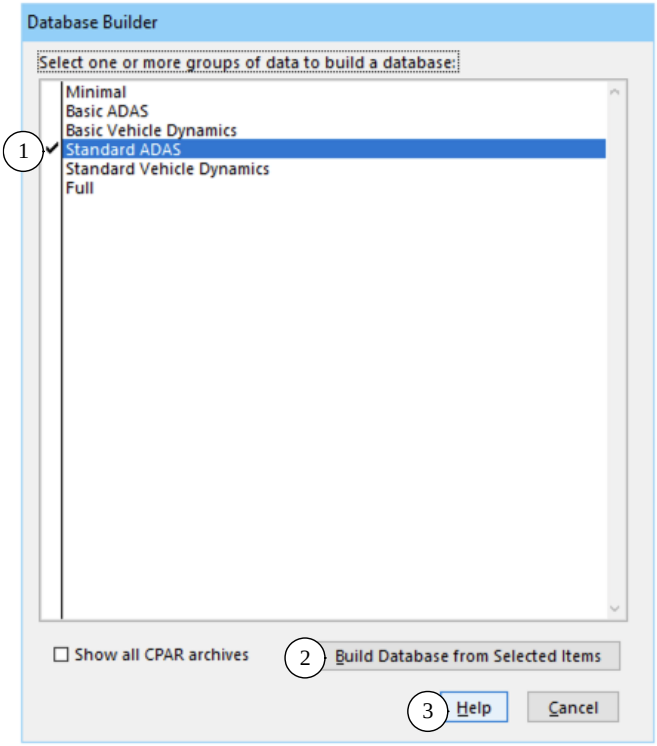

Figure 2. The database builder window showing only predefined database options.

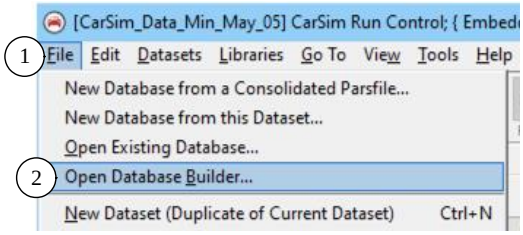

Figure 3. Use the File menu command Open Database Builder.

Predefined database options

Figure 2 shows that CarSim 2021.1 provides six predefined databases. Table 2 provides a little more information about these options.

Specialty Examples

None of the six basic options shown in Figure 2 include specialty examples, such as RT HIL systems, software development kit (SDK) examples, examples with trailers or engine mounts or frame twist, examples using external tire models, etc. To access these, and other specific categories of examples, check the box **Show all CPAR archives** (3) (Figure 4) to see the entire collection.

Notice that many CPAR archives are available, shown below the six database options described in Table 2.

Table 2. Pre-defined database options.

Database Option	**Description**	**Target Users**
Minimal	Examples showing new features, plus core examples such as the Quick Start Guide example, output options, payloads, validation examples, and Preferences	Experienced users with existing databases
Basic ADAS	Minimal + most ADAS examples	New users interested in basic ADAS options
Basic Vehicle Dynamics	Minimal + examples showing vehicle properties and tests	New users interested in basic vehicle dynamics options
Standard ADAS	Basic ADAS + driver model examples, multiple vehicles, roads, proving grounds, LabVIEW, Simulink, VS Visualizer data	Users interested in all examples relevant for ADAS applications
Standard Vehicle Dynamics	Basic Vehicle Dynamics + driver model examples, roads, Proving Grounds, LabVIEW, Simulink, K&C Sequence, vehicle configurations, VS Visualizer data	Users interested in all examples relevant for vehicle dynamics evaluations
Full	Standard ADAS + Standard Vehicle Dynamics + Advanced VS Commands, Custom Forces and Motions, Embedded Python, Extended Models, External Control, Kinematical Preview. Like example databases from older versions.	Users interested in all capabilities that do not require extra licenses or software (other than Sensors, Simulink, or LabVIEW)

Table 3 lists the specialty CPAR archives that are not included in any of the six pre-defined groups. For more information, including the descriptions of all CPAR archives, please see the Help document for the Database Builder.

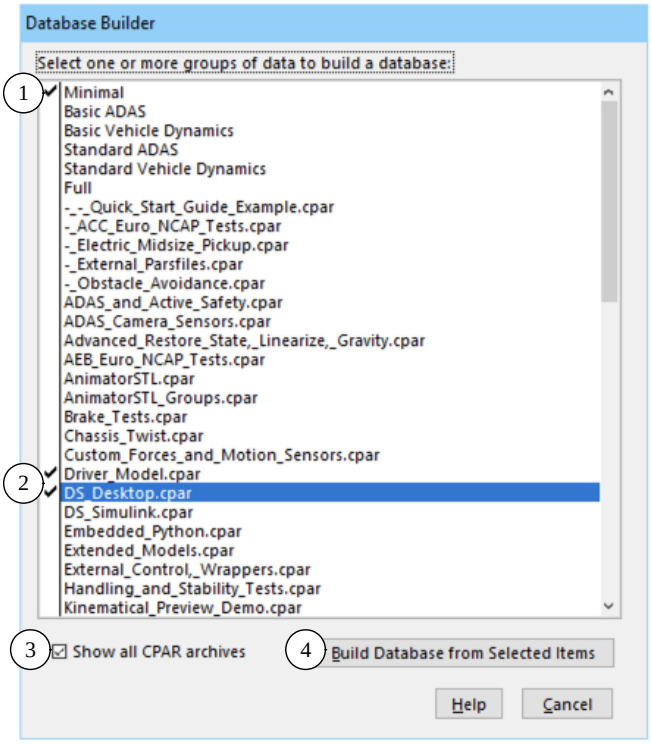

Figure 4. Display when CPARs are also shown.

Table 3. Specialty CPAR archives for standalone use or that require special licenses.

Category	**Description**
ADAS: Camera Sensors	Use sensor memory shared by VS Visualizer
Chassis Twist	Examples showing the flexible body option
DS: Desktop	Driving simulator examples: stand-alone
DS: Simulink	Driving simulator examples with Simulink
Powertrain (External Software)	Examples with AVL Cruise and GT Suite
Powertrain Mounts	Examples with powertrain mounts
RT: A&D	Examples for A&D RT system
RT: Concurrent	Examples for Concurrent RT system
RT: dSPACE	Examples for dSPACE RT systems
RT: ETAS	Examples for ETAS COSYM and LCO systems
RT: NI	Examples for NI ETS and LinuxRT systems
RT: RT-Lab	Examples for RT Lab system
SAE: *name (7 categories)*	SAE examples for FSAE and Baja
SDK: Extended Models	Datasets for SDK examples that extend models or provide alternative wrappers
SDK: External Control	
Tire Models (External Software)	Ftire, MF_Swift, TameTire
Trailers (Extra License Needed)	Example using vehicles with trailers
VI-DriveSim Integration	Example for running in VI-DriveSim

More information about specific new dataset categories is provided in the last section of these release notes (page 27).

GUI Support for New Solid Axle Features

Most of the new features for the solid axle suspension model (described in the following VS Math Model section) are supported in the GUI:

1. Updated **Suspension: Roll Steer (Solid Axle)** library screen, to support 2D table types and combinations of 1D tables.
2. New **Suspension: Dive Angle (Solid Axle)** library screen, to support 2D table types and combinations of 1D tables. The **Suspension: Dive Angle (Caster Change)** library screen is no longer recommended for solid axle suspension types but will still work with legacy datasets.
3. New **Suspension: Longitudinal Movement (Solid Axle)** library screen, to support 2D table types and combinations of 1D tables. The **Suspension: Longitudinal Movement** library screen is no longer recommended for solid axle suspension types but will still work with legacy datasets.
4. Updated **Suspension: Lateral Movement Due to Roll and Jounce** library screen, adding a checkbox for the optional lateral definition enabled by `OPT_SUSP_Y_AXLE_ROLL`. As a reminder, this screen already supported 2D table types and combinations of 1D tables. It is the recommended solid axle lateral movement screen.
5. Updated **Suspension: Solid Axle System Kinematics** screen, to enable linking to the new dive and longitudinal movement libraries. Non-recommended dive, lateral, and longitudinal movement links will be detected and shown as orange instead of blue upon refreshing this screen. Additionally, an empty dataset created using this library screen will now default to the recommended link types for the dive angle, longitudinal movement, lateral movement, and roll steer tables.

Road: Animator Repeated Object Screen

This screen contained a feature in which objects can be made detectable by sensors. New input fields have been added which allow users to specify the `Lx_Front` and material type of the detected objects.

Tire (External) Screen

There are some enhancements and cleaning up of the Tire (External) screen to accommodate changes in the external tire models, such as discontinued TNO models that have been replaced by Siemens.

1. Removed **TNO MF-Tyre v6.2** and **TNO MF-Tyre/MF-Swift v6.2** model options as well as all corresponding model solvers, libraries and supporting files for the TNO tire model.
2. Added **Siemens MF-Tyre Only** option which distinguishes all features that can run with a CarSim license on Windows. Any feature item appearing in **Contact**, **Dynamics**, and **Slip forces** with this MF-Tyre only model option does not require a separate license from Siemens for Windows.

3. Datasets with discontinued model options (**TNO MF-Tyre v6.2** and **TNO MF-Tyre/MF-Swift v6.2**) from prior versions can be imported and automatically assigned to either **Siemens MF-Tyre Only** or **Siemens MF-Tyre/MF-Swift** model options.
4. External road data file (RDF/CRG format) is now selectable with the Siemens model.
5. Added **Michelin TameTire** model option which enables you to specify the tire property file (TIR), TameTire model solver for Windows (DLL), RT library (SO file for dSPACE SCALEXIO) and Windows wrapper (VS STI module DLL). The tire property file is automatically converted to a VS format table. The file conversion requires a separate license from Michelin.
6. The RT Library field can specify not only the library for DS1006 (LIB) but also for Concurrent RT and dSPACE SCALEXIO (SO).

Support FMU with External Tire Model

The FMU option has been extended to include support for the external tire models MF-Tyre from Siemens and FTire from COSIN. These require that FMU 2.0 be specified.

New Time and Version Stamps Added to Parsfiles

Parsfiles written by a VS Browser are named using a 36-character unique universal ID (UUID), with a prefix associated with the library, e.g., `Run_6cb07365-85a3-4a3b-95f9-c5c844687601` is a name that might be used for the Run Control library. This is done to support version control software that tracks creation and revisions of files based on their names.

The Parsfiles always include stamps at the end of the file beginning with '#', such as `#Library`, `#Dataset`, `#Category`, `#fileID`, and `#Product`. Three new stamps were added in 2021.1 to provide more information about time and version. Figure 5 compares the content at the end of two similar files made in versions 2021.0 (left side) and 2021.1 (right side). Both are copies of the same file: the Baseline run used for the Quick Start Guide example. The `#FileID` names differ, as expected, because they depend on UUIDs created uniquely for each file. The `#Product` stamps differ, also as expected, because two different versions were used to create and view the files.

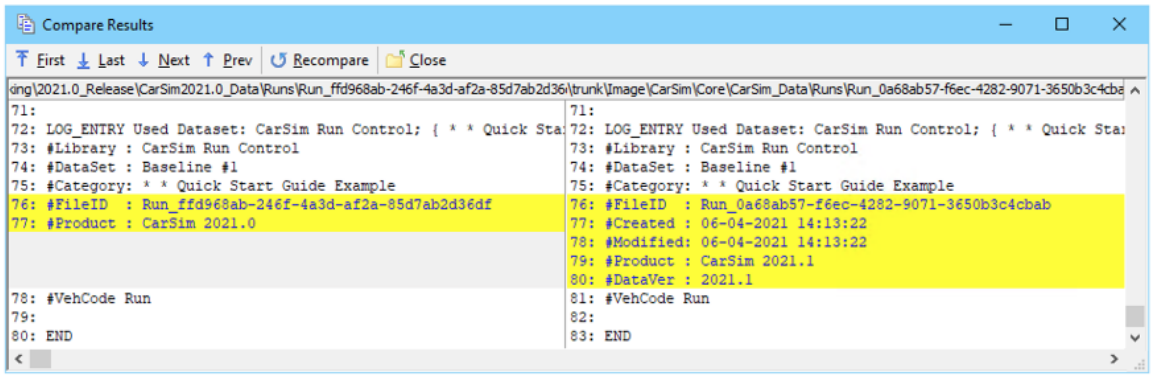

Figure 5. Timestamps compared for versions 2021.0 (left side) and 2021.1 (right side).

Note that the newer version has three new stamps:

1. #Created identifies the date and time (obtained from the computer OS) when the file was created. This stamp will never be changed by the Browser even after updates to newer versions.
2. #Modfied identifies the date and time (obtained from the computer OS) when the file content was last changed. If a change is made, the time-date from the #Modified stamp is increased. However, if a change is made and then the Undo command is used (Ctrl+Z or the Edit menu), the file contents are restored, and the time-date value is also restored to the value it had before the change was made.
3. #DataVer identifies the version of the database. With all recent releases, the Database version (#DataVer) has matched the product version (#Product). However, this has not always been the case. (Sometime minor updates have been made for the product, but the database version was not changed.)

VS Math Models

S-L from X-Y Coordinate Transformation

Two new options are available for the potentially iterative calculation of S-L (station and lateral coordinate) from X-Y path coordinates during the simulation:

1. OPT_SL_METHOD (zero or one). When OPT_SL_METHOD=1, the estimate for station at any given iteration includes a dependence on path curvature. This can lead to more rapid convergence to a solution for station if the curvature is well-behaved. If the VS Math Model detects a failure to converge, the solution is re-attempted, this time not using path curvature. This is done automatically. When OPT_SL_METHOD=0, curvature is never used (the attempt using curvature is bypassed entirely), reproducing the old behavior. The default setting as of 2021.1 is OPT_SL_METHOD=1.
2. TOL_SL_METHOD, to fine-tune the accuracy of the S-L from X-Y iteration. The iteration halts when the distance is less than or equal to TOL_SL_METHOD, which has units of millimeters. This existed in the VS Math Model previously but was not settable by the user. The old value was 5 millimeters. The default setting as of 2021.1 is 1 millimeter.

In our testing, the new default settings (OPT_SL_METHOD=1, TOL_SL_METHOD=1 mm) maintain or slightly improve performance compared to the old behavior (OPT_SL_METHOD=0, TOL_SL_METHOD=5 mm), while offering improved accuracy.

New 2D Suspension Tables

Various tables in the suspension models have been updated to support 2D table types.

1. The solid axle roll steer, math model keyword SUSP_AXLE_ROLL_STEER, can now be specified with a secondary independent variable of axle jounce. The primary independent variable remains axle roll (roll relative to axle). The 2D effect can be given directly with a 2D table type or combined from two 1D tables. The secondary 1D table uses keyword SUSP_AXLE_JNC_STEER.
2. The solid axle dive angle, math model keyword SUSP_DIVE_AXLE, can now be specified with a secondary independent variable of axle roll. The primary independent variable

remains axle jounce. The 2D effect can be given directly with a 2D table type or combined from two 1D tables. The secondary 1D table uses keyword SUSP_DIVE_AXLE_ROLL.

3. The solid axle longitudinal movement, math model keyword SUSP_X_AXLE, can now be specified with a secondary independent variable of axle roll. The primary independent variable remains axle jounce. The 2D effect can be given directly with a 2D table type or combined from two 1D tables. The secondary 1D table uses keyword SUSP_X_AXLE_ROLL.

4. The CarSim virtual steering axis suspension now supports 2D tables of jounce and steering rack travel for the jounce and rebound stop kinematics (new VS Math Model keywords CMP_JSTOP_VIR and CMP_RSTOP_VIR). Previously, these could be entered only as 1D tables of jounce. The additional relationship with rack travel allows the jounce and rebound stops to potentially contribute to the rack force from the tie rod. The two example vehicles in CarSim that use the virtual steering axis — *B-Class Sports Car (VSA)* and *D-Class Sedan (VSA)* — have been updated to include 2D jounce and rebound stop data.

5. The non-virtual steering axis suspensions now support 2D tables of jounce and jounce other side for the spring and damper compressions (VS Math Model keywords CMP_SPR_SEAT and CMP_DAMP, respectively). Previously, these only supported 1D functions of jounce. This is most useful for the 2018 twist beam model, which uses the independent suspension with 2D kinematics tables. The example vehicle for the 2018 twist beam model has been updated to use the 2D table type for these.

6. The non-virtual steering axis suspensions now support 2D tables of jounce and jounce other side for the jounce and rebound stop kinematics (VS Math Model keywords CMP_JSTOP and CMP_RSTOP, respectively). Previously, these only supported 1D functions of jounce. The example vehicle for the 2018 twist beam model has been updated to use the 2D table type for these.

Miscellaneous

1. New output variables were added to provide real-time performance: T_Real_Elapsed (real clock time since start of run), T_Real_Step (real clock time used by the VS Math Model to perform calculations for the last time step, but not counting time used by external software such as Simulink), and T_Real_Last (total real clock time used for the last time step by the VS Math Model plus external software).

2. Users can now use VS Commands to define indexed variables and parameters with up to two dimensions. Previously, only indexed variables or parameters of one dimension could be defined by the user.

3. The solid axle lateral movement table, math model keyword SUSP_Y_AXLE_ROLL, supports a new option controlled by keyword OPT_SUSP_Y_AXLE_ROLL. When OPT_SUSP_Y_AXLE_ROLL=1, the solid axle lateral movement table completely specifies the kinematical lateral displacement of the axle center point relative to the sprung mass. When OPT_SUSP_Y_AXLE_ROLL=0, the solid axle lateral movement table contributes partly to the kinematical lateral displacement of the axle center point relative to the sprung mass; there is also a lateral contribution from the axle jounce in the direction implied by axle roll. The setting of OPT_SUSP_Y_AXLE_ROLL=0 is intended for

backwards compatibility with legacy datasets. For new datasets, we recommend turning on the new option using the new checkbox (see notes on new GUI features below).

4. Solid axle suspension types now support a nonzero pitch inertia for the unsprung axle body, math model keyword `IA_YY`. This may be set from a miscellaneous yellow field.

5. Solid axle suspension types now support a nonzero longitudinal CG offset for the unsprung axle body, math model keyword `X_CG_AXLE`. This allows the axle CG's longitudinal location to differ from that implied solely by the wheelbase. This may be set from a miscellaneous yellow field.

6. When the unsprung mass contribution to steering system inertia is turned on (`OPT_I_GEAR_IN=1`), the instantaneous, total steering system inertia value is now printed in Echo files as `I_GEAR_IN_TOT`. The instantaneous inertia value remains available for plotting via the output variable `IstrGear`.

7. A proper error message is added for the case that "Motors on driven axles" is selected with hybrid/electric powertrain but no motor is selected on the differential screen.

RT Platforms

Starting with version 2021.1, CarSim supports COSYM v2.3 from ETAS with Simulink or FMI 2.0 for COSYM and parallel solvers application.

VS Visualizer

VS Visualizer can now compute first and second derivative estimates of data channels and use these values for both animation and plotting. See the VS Visualizer reference manual for more information.

Documentation

The following documents have been added to the **Help** menu:

1. CarSim Database Options (Release Notes)
2. CarSim Known Issues for Converting Old Databases (Release Notes)
3. Euro NCAP ACC Tests (Technical Memos)
4. VS Table Tool Introduction (Reference Manuals)
5. VS Table Tool Usage (Technical Memos)
6. Windows DS for CarSim and TruckSim (Real-Time and DS Systems)

The following Guides and Tutorials have been updated:

7. * Introduction to CarSim
8. Borrowing a License from a Network Server
9. CarSim Demo Tutorial
10. CarSim Quick Start Guide

The following Deprecated Items have been updated:

11. Clutch Control (Closed Loop) Screen
12. Internal Pacejka 5.2 Tire
13. Path X-Y Coordinate Screens
14. Powertrain Transmission Screens
15. Speed (Closed Loop) vs Station Screen

The following Reference Manuals have been updated:

16. MF-Tyre/MF-Swift User Manual
17. VS Browser (GUI and Database)
18. VS Math Models
19. VS COM Interface
20. VS Commands
21. VS Commands Summary
22. VS Visualizer

The following Screen documents have been updated:

23. ADAS Sensors and Moving Objects
24. Animator Reference Frames
25. Animator Shapes and Groups
26. Animator Wheel Arrows and Other Indicators
27. Batch Matrix
28. CarSim and TruckSim Suspensions
29. CarSim Steering Systems
30. Driver Controls
31. External Models and RT Systems
32. Paths and Road Surfaces
33. Road Surface Visualization
34. Run Control Screen (Home)
35. Tire Models

The following Technical Memos have been updated:

36. Automating Runs with the VS API
37. Convert Simulink Model into Executable

38. Example: Extending a Model with VS Commands and the API
39. Extending a Model with Embedded Python
40. HPC Licensing
41. Parking Lot Layout and Dimensions
42. Simulation with Multiple Vehicles
43. Twist Beam Suspensions: Using 2D Tables
44. VS Solver Wrapper

The following Real-Time and DS System documents have been updated:

45. Concurrent RT Guide
46. NI RT Guide
47. VI Integration Guide

The following Software Development Kit (SDK) documents have been updated:

48. The VehicleSim API — Running and Extending VS Solvers
49. The VS Connect API — Inter-process Data Replication via UDP

Database

Database Builder

With the new Database Builder, each category in the Run Control library is provided with a separate CPAR archive file. Many of the categories and titles of existing database were modified. If the content is otherwise the same, then the original UUID file name was kept.

In many of the categories, more runs were included to ensure that older datasets of interest would still be included. The following subsections describe significant new examples or updates of existing examples.

New and Updated Examples

Solid Axle Examples Use New Recommendations for Library Screens

The solid axle suspension examples have been updated to use the new **Suspension: Dive Angle (Solid Axle)** and **Suspension: Longitudinal Movement (Solid Axle)** library screens**.**

Euro NCAP ACC Performance Examples

These CarSim Euro NCAP ACC Performance examples are meant to show users how to implement their own ACC logic and tuning parameters within the CarSim environment, to better understand how their ACC will perform against the Euro NCAP ACC Performance tests. There are 11 examples from the Euro NCAP ACC Test and Assessment Protocol including new Cut-In and Cut-Out examples. Additionally, there are two non-Euro NCAP examples included which are meant to aid in basic ACC testing, before users attempt the more complicated Euro NCAP testing.

Online Path Update/Obstacle Avoidance Examples

This example a passenger car on a path that become obstructed with an obstacle, then uses an onboard autonomous path update procedure to avoid the obstacle, and return to the original path, as a double lane change (DLC). Simulink is used in this example, to create an updated path, and provide updated driver model preview point coordinates, such that the steering controller adjusts the steering to avoid the obstacle. The goal of this example is to show how to change the vehicle path traveled while after the initial path Dataset is loaded. This example also shows how to execute the key steps in the 'Update Path with Embedded Python' example, only using Simulink instead of Python and the VS API. There are two examples in this new Obstacle Avoidance Category:

- One utilizes the same spline trajectory as the 'Update Path with Embedded Python' example as a double lane change (DLC),
- The other uses a high-order polynomial trajectory as a single lane change (SLC) in path update mode, which then switches back to the CarSim driver model to bring the vehicle back to the original example path.

New Midsize Pickup with Fully Electric Powertrain and 2D Solid Axle Kinematics

The new solid axle features (2D kinematics, pitch inertia, and longitudinal CG offset) are demonstrated with a new fully electric midsize pickup example, including several different test procedures, located in the "* *Electric Midsize Pickup"* category, part of the minimal database collection. The 2D kinematics data were gathered from a SuspensionSim example of a four-link solid axle suspension with track bar. The electric powertrain is dual-motor, with a 93 kW-h battery pack having a nominal voltage of 723 V.

Exotic Sports Car: Powertrain Data Updates

CarSim ships with two versions of the Exotic Sports Car: one with a 4WD powertrain performing some acceleration tests, and a RWD race car-type version on the Handling Course. The 4WD version received some data updates over the past few releases that had the unintended consequence of preventing it from completing the Event-based Full Throttle Acceleration test.

For the 2021.1 release, this vehicle has received updates in the form of a new value for the driveline frequency (`DRIVELINE_FREQ` is now 9 Hz vs. 12 Hz) and a 7-speed transmission (vs. the previous 6-speed) with new gear ratios and shift schedules. Acceleration and coast-down tests were performed to evaluate the new data over a range of throttle and road grade inputs.

The corresponding Event-based Full Throttle Acceleration test has also been revised to achieve a more realistic launch sequence. The throttle ramp to 100% has been slowed down, as has the clutch release. The brakes, initially applied, are released smoothly. Once the vehicle is rolling and a specific driveline speed has been reached, Events are used to switch from Open Loop Transmission Control to Closed Loop Transmission Control.

External Parsfiles

The VehicleSim GUI has a screen within the Generic library category called External PARSFILE. This allows an end-user to link to a parsfile that they create themselves. Once the user-generated parsfile is linked to this screen, the contents of the Parsfile will be provided as input to the VS Math Model if a link is made to the dataset.

In response to questions from many end-users regarding how to link to a parsfile that they have created — i.e., mainly road and path data — the 2021.1 version of CarSim includes two **Run Control** examples that demonstrate the use of the External PARSFILE library.

Updated AEB Example

The automatic emergency braking example *AEB CCRb w/ Deceleration Command*, which uses the speed controller in target acceleration mode, `OPT_SC=5`, to execute the emergency stop, has been updated to prevent conflicting speed controller settings between the vehicle and the procedure.

Added More Powertrain Mount Examples

Five new powertrain mount examples were added, including torque-tube, mount sensors, and hybrid e-AWD.

Miscellaneous Examples

- VS STI simple tire examples are changed to be more robust and showing a proper warning message especially when STI_TYPARR table is not read.
- VS STI wrapper for FTire model is changed to be more robust and showing a proper warning message especially when an unsupported version of FTire (older than v2020-4) is used.