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


# CarSim 2023.1 New Features



This document lists notable new features in CarSim 2023.1.

## 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.

### *PID Controllers*

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.

### *Numerical Differentiation*

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.

### *Modes for Import variables*

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 divide*

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

### *Defining Import and Output variables*

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.

### *Calling the vs_statement API function*

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.

### *Callback locations*

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 Tutorials were updated:

3. Running a VS Math Model in Simulink

The following Reference Manuals have been 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 have been 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 have been 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 documents have been updated:

28. dSPACE RT Guide

The following SDK documents have been 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' in order 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.