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

Technical Memo
carsim.com

CarSim 2018.0 Release Notes

Architecture Changes

Modularity

In all prior versions of CarSim, custom math models were created for multibody models of interest, such as a motor vehicle with a front independent suspension and solid-axle rear suspension towing a two-axle trailer (both solid-axle suspensions), with a code such as

I_S__SS. CarSim 2017 included custom solvers for 36 distinct multibody models. On Windows, each model was handled with separate solvers for 32-bit and 64-bit operation; hence there were 72 CarSim solvers.

Version 2018 introduces the outcome of a four-year project to redo the architecture of VS Solvers to support more user-selectable options. New methods for connecting multibody models were developed that maintain the fidelity and computational efficiency of past versions, while supporting the options to install parts of the model at runtime. The result is that CarSim 2018.0 includes just two Windows vehicle solvers: carsim_32.dll and carsim_64.dll. Each solver includes modules for all types of motor vehicles, trailers, and suspensions that were supported in past versions. The modules are used only if needed for each simulation.

The number of solvers is also reduced for Linux and all real-time systems. Most have a single solver; those that support both 32-bit and 64-bit dynamic libraries include one solver for each.

Using version 2018 is the same as using prior versions for most activities. The same parameters, state variables, outputs, imports, etc. exist in the new version, with the same keywords, user interface, and datasets. The Echo file is not affected by the modularity.

The differences that will be seen with this release are:

1. New suspension options are available for trailers.
2. The hitch model was changed, resulting in minor differences when a trailer is included.
3. Some models run faster, especially those with trailers.

More details about the changes related to modularity are provided in the memo *Modular Vehicle Models in Version 2018*, available from the Help menu: **Help>Release Notes>CarSim 2018: Modular Vehicle Models**.

Longer Names for Output Variables

The VS output file format was introduced in 2014 as an alternative to the legacy ERD file format. In version 2017, support was added for comma-separated variable (CSV) text files that can be opened by Excel and other spreadsheet programs. ERD files restrict the short variable names used as keywords to 8 characters; CSV and VS files support longer names.

The VS Solvers in the 2018 release make use of longer names in order to provide consistent naming of variables when there are many instances (e.g., more than nine ADAS sensors). If the output variable has a name longer than 8 characters, VS and ERD output files include both the long name and a name limited to 8 characters. VS Visualizer and the VS/ERD File Utility now recognize both the SHORTNAM and the NAME keywords when reading ERD files. Therefore, datasets for plots and animation from older versions that specify names with the 8-character limit are processed correctly.

The keyword SHORTNAM has been used since the ERD format was published in 1984. Because the SHORTNAM information is still written to the ERD header, legacy third-party utility programs that support the ERD format should still operate correctly.

Documentation files that list outputs (as obtained with the **View** button on the **Run Control** screen) show both full and short names when the full name has more than 8 characters, as do the

VS Browser screens used to view and select outputs, such as **Plot Setup**, **I/O Channels: Export**, and **I/O Channels: Write**.

Binary Table Files

Tabular data used in Configurable Functions have traditionally been provided in text Parsfiles. In some cases, very large tables are used to represent road surface properties. A result is that the time needed by the VS Solver to read the input Parsfiles can be excessive. For example, some datasets used in driving simulators can take more than 20 minutes to load when the tables are provided in text form.

CarSim 2017 included some example simulations that took place in Mcity, with the ground geometry provided by 2D tabular data read from an ERD binary file. In this case, the time to load the binary data is about 0.15s, reduced from about 20 minutes for a text version.

In version 2018, an API (`vs_table.dll`) has been developed to support this type of usage, with utilities for creating binary table files (`.vstb`) that are well-suited for reading large tables in a short time.

An API (`vs_output.dll`) has also been developed to support the reading and modification of VS Output files (`.vs`) that will aid the user in modification of VS Outputs for use by other programs or processes.

Full Support for User-Defined Differential Equations

VS Commands were introduced in CarSim in 2007, including the option to add ordinary differential equations (ODEs) with the command `EQ_DIFFERENTIAL`. The ODEs added with VS Commands have been numerically integrated with the same methods as the built-in ODEs. However, some features were not fully supported.

Starting in version 2018.0, user-defined state variables are listed in machine-generated documentation files (accessed with the **View** button on the lower-right corner of the **Run Control** screen), the same as built-in state variables.

User-defined state variables are also listed in the End Parsfile (along with built-in state variables) to support a new run that continues from an existing simulation run.

Error and Range Checking in VS Command functions

Formulas used in VS Commands and other inputs can make use of about 45 built-in functions. In past versions, functions such as `sqrt`, `pow`, `sin`, `cos`, were handled by passing the arguments directly to native C library functions (as declared in `math.h`). A few of these functions have range limits. For example, the function `asin` requires that the argument be within ±1; the function `sqrt` requires that the argument not be negative.

In version 2018, two levels of range checking are added. Functions that have known ranges now check the argument. If it goes outside the range, the argument is changed to the limit and then the function is applied. For example, `asin(-1.01)` is evaluated as `asin(-1.0)`. As another example, `sqrt(-0.123)` is evaluated as `sqrt(0.0)`.

If results show a math exception (`nan` or `inf`, indicating “not a number” or “infinite,” respectively), an error is generated, and the formula is shown.

In past versions, the formula in the VS Command would return a `nan` or `inf`, but there would be no reporting. In many cases the VS Commands were not critical, so users were unaware of the issue.

Writing Output Variables for Optional Parts

CarSim has many output variables that are associated with the basic parts of the multibody model, such as sprung mass motions, tire kinematics and forces, suspension deflections and forces, etc. These variables always exist, and can be plotted, applied to animate auxiliary objects (e.g., force arrows), and used in VS Commands. The variables are activated for writing to file by referring to them in datasets from the **Plot Setup**, **Animator Reference Frame**, and **I/O Channel Output** libraries.

The math model also has many parts that are optional (e.g., reference points) and also have associated output variables. The output variables for the optional parts are added to the model when the parts are defined automatically by linking to datasets that define the properties of the parts.

The default behavior for VS Commands that create new output variables was changed in v2018 to define the variables, but not activate them for writing to file. When the checkbox “Write all outputs” on the **Run Control** screen is not checked, outputs are activated for being written to the output file only if they are used in animations or plots. Of course, they still exist, and may always be used in VS Commands and/or exported to other software.

In some ADAS scenarios, there can be tens of thousands of output variables associated with moving objects and detections by ADAS sensors, so this behavior could result in very large output files. The new behavior supports the creation of much smaller files, and sometimes faster simulation.

Model Features

New Vehicle Configurations Supported by Modularity

As noted earlier, CarSim 2018.0 includes just two Windows vehicle solvers: `carsim_32.dll` and `carsim_64.dll`. These support all vehicle configurations that were possible in prior versions, plus configurations that were never available in earlier versions. For example, trailers can have any combination of suspension types.

The CarSim GUI supports a motor vehicle lead unit with two suspensions and a trailer with up to two suspensions. The front axle is a steered independent or solid-axle suspension. The rear axle of the motor vehicle may be steered or un-steered, and may be independent, solid axle, or legacy twist beam. The trailer suspensions may be un-steered independent or solid axles.

Advanced users can also simulate trailers with three axles, with tridem load sharing by setting up the trailer configuration using generic screens.

Details of the modules includes in CarSim are provided in the memo *Modular Vehicle Models in Version 2018*, available from the Help menu: **Help>Release Notes>CarSim 2018: Modular Vehicle Models**.

Vehicle Dynamics

Steering system

The steering system supports column-assisted power steering. The assist torque is applied to the pinion or input gear.

Brakes

The wheels on the vehicle now include a bearing friction (keyword = `MY_FRICTION`) that is automatically added to the brake torque applied to the wheel. The friction torque is applied even when the brake system is not in use, providing more realistic behavior in scenarios where the vehicle is at rest and neither the brake system nor powertrain are connected.

Generic suspension model

Independent suspensions have been extended to include 2-D options for 5 kinematics and 9 compliance tables, with the two independent variables being the jounce on the current and opposing side. This extension supports K&C data obtained with twist-beam suspensions and possibly some other types of suspensions.

The suspensions have been renamed "generic" because the two sides of the suspension are not completely independent when the 2-D tables are used. However, the vehicle code still uses the letter '`I`', as in the code `I_S` for a vehicle with an independent (generic) front suspension and a solid-axle rear suspension.

Powertrain

Shift duration can now be set separately for upshifting and downshifting of an automatic transmission, with different values for each gear position (new keywords: `T_SHIFT_UP` and `T_SHIFT_DOWN`). In support of backward compatibility, the default values for the new durations are –1, in which case the default duration is used (the default duration keyword is still `T_SHIFT`).

Another improvement in the powertrain model is that the AVL Cruise screen now supports up to 18 gears shift schedules (it was previously limited to seven gears shift schedules).

Tires

Effective rolling radius is now extendable from a constant (specified on the Tire screen) to a time variable by importing the data from VS Commands, Simulink, etc.

Two tire kinematics outputs for each tire (`YawWK_L1`, `RollWK_L1`, …) were added. They are the Euler yaw and roll angle of the wheel knuckle (the Euler sequence is yaw->pitch->roll), which are inputs to the VS STI tire models.

Suspension output variables

Three suspension kinematics outputs for each suspension side (Dive_L1, SusX_L1, SusY_L1, …) are added. They are the wheel knuckle dive angle, wheel longitudinal displacement, and lateral displacement from the design location based on the sprung mass coordinate system and can be useful to correlate with real K&C test data.

Vehicle configuration parameters

Information about the vehicle configuration is available for use in VS Commands. Properties are written in the Echo file with keywords VEHICLE_NAXLES (total number of axles), VEHICLE_NUNITS (total number of vehicle units), and VEHICLE_AXLE_MAP(*u*) (number of axles for vehicle unit *u*). These values are read-only.

Aerodynamics

Aerodynamic slip is the angle between the absolute wind vector and the vehicle velocity vector. The point in the vehicle used to define the velocity vector was changed from a point fixed in the sprung mass to the instant location of the total mass of the lead unit. This was the original intent years ago, but the methods used to build the model required an approximation.

The ground reference used for aerodynamic jounce and pitch was previously based on a single point on the road surface under the aerodynamic reference point fixed in the sprung mass. It is now calculated using the contact point elevations for the four tires.

GPS Outputs

Output variables for GPS latitude and longitude in past versions were based on parameters for reference latitude and longitude for global X-Y coordinates of (0,0). Three new parameters were added for version 2018 in support of large changes in global Y coordinates. These are reference X and Y coordinates corresponding to the reference GPS latitude and longitude, and a limit of Y change that will trigger an automatic GPS recalibration.

Paths and Roads

The maximum number of reference paths was increased from 100 to 200. The maximum number of supporting tables such as LTARG (lateral offset) and SPEED_TARGET (target speed for ego vehicle or moving object) was also increased to 200.

When working with external software that provides road surface information, there was previously a check to ensure that all information was provided. For each point of interest, the import variables are Zgnd, Mux, Muy, Dzdx, and Dzdy. The error-checking has been removed in support of setups where Zgnd is provided but other variables are intended to keep internal values.

VS SNAPI

The new VS SNAPI Communication Library has been added to the folder CarSim_Prog\Programs\vs_snapi. VS SNAPI is a DLL with a public C API header file. SNAPI can be used to facilitate communication between multiple processes on one or more computers. For example, the SNAPI example Simulink model and S-function (located in

`Extensions\Simulink\SnapiExample`) uses SNAPI to communicate with a separate remote process that contains a CarSim Solver. The forthcoming release of the CarSim Vehicle Dynamics plugin for Unreal Engine (https://www.unrealengine.com/marketplace/carsim-vehicle-dynamics) will include embedded support for SNAPI, allowing the CarSim vehicle model to run within the Unreal Engine environment while utilizing external controllers running in Simulink.

ADAS (Sensors and Moving Objects)

Many improvements have been made involving the ADAS sensors and moving objects.

Optional writing of output variables

In some ADAS scenarios, there can be tens of thousands of output variables, so this behavior could result in very large output files. As mentioned earlier, the default behavior was changed for version 2018; the outputs are not activated by default. When the checkbox “Write all outputs” on the **Run Control** screen is not checked, outputs are typically activated by using them in animations or plots. This means the output files for ADAS simulations in version 2018 may be significantly smaller, and the simulations can run faster.

Moving objects

The maximum allowable number of moving objects was increased from 99 to 200.

More support has been added for using moving objects to represent traffic vehicles. A speed variable `V_OBJ_`*o* has been added to provide a consistent differential equation for calculating station. A parameter `OPT_SPEED_OBJ` was added with three options for handling speed: 0 → don’t use speed to calculate station or X-Y coordinates, 1 → integrate `V_OBJ_`*o* to obtain station, and 2 → integrate `V_OBJ_`*o* to obtain station and also use a new wheelbase parameter `L_WB_OBJ` to calculate heading angle with a differential equation that simulates vehicle offtracking. When offtracking is enabled, the motions of the moving objects resemble the behavior of real vehicles even in severe low-speed turns. When `OPT_SPEED_OBJ` is zero, the variable `V_OBJ_`*o* is calculated using finite differences.

X and Y coordinates were added as output variables for the four corners of rectangular objects. For cylindrical objects, the four points are for the front, right, rear, and left.

GPS latitude and longitude were added as output variables for all moving objects.

Information about an object shape is always used if there is at least one ADAS range and tracking sensor in the simulation. A parameter named `OPT_SHAPE_OBJ` was added to also use shape information for moving objects in case there are no ADAS sensors.

ADAS sensors and detections

The maximum allowable number of ADAS sensor was increased from 20 to 99.

A range-limit parameter `RANGE_LIMIT_SENSOR` was added to the sensor model. It is used to eliminate calculations for moving objects that are guaranteed to be out of range for a sensor; this can reduce the simulation time noticeably when there are many objects and/or sensors.

A similar method is used to eliminate moving objects based on bearing angle, using the existing field of view (FOV) parameters.

Another parameter was added for sensors in general. When the new parameter `OPT_ALTERNATE_SENSORS` is given a non-zero value, only one sensor is updated each time step. The sensor being updated alternates among all sensors. For example, if there are four sensors, each one is updated every fourth time-step. In nearly all cases, the parameter can be safely given a nonzero value; the default value is 1. When the value of this parameter is zero, all sensor detections are calculated every time step, as was done in past versions.

The new parameters `RANGE_LIMIT_SENSOR` and `OPT_ALTERNATE_SENSORS` can speed up the simulation, with the difference being more significant for large numbers of moving objects and sensors. For a test case with 20 sensors and 99 moving objects, the calculation decreased by a factor of 10 (90% reduction in computation time)!

The connection to the main VS Solver was redone for 2018 to provide better support for VS Commands and Events. Past versions did not support changes to path or road ID numbers after the run initialized; version 2018 provides fully control for changing properties with events. In past versions, initialization was not complete at T=0; now it is.

Output variables for detections all involve the connection between one sensor and one moving object. Each time a sensor or object is created, output variables for all potential detections are created. In past versions, output variables for detections could not be used in VS Command equations until both the sensors and moving objects had been defined. CarSim 2018 adds a new sensor parameter `N_MIN_DETECTIONS` that causes the `DEFINE_SENSORS` command to create a minimum number of sets of detection output variables. If left at the default value of 1, each time a sensor is defined, the output variables for one detection will also be created. This feature enables sensors to be created before moving objects, as is the case in the CarSim GUI if the sensor datasets are linked to a vehicle dataset.

New detection output variables were added: the global X and Y coordinates for the left and right edges of a detected object.

A new algorithm was implemented to control occlusion of objects. In version 2018, a parameter for each sensor `OPT_LEGACY_SENSOR` can be set to 1 to disable the new algorithm (the default is 0). This new algorithm provides increased accuracy in certain edge cases. These edge cases are showcased in the "`* CS 2018 – Sensor Improvements`" category. The yellow field notes panel for each of the three examples gives a detailed explanation of the improvements being demonstrated.

VS Visualizer

The VS Visualizer Application

A new option **Export Video** has been added (available from the **File** menu) to create an AVI (audio-video interleaved) file that can be dragged into Microsoft PowerPoint® or viewed with Windows Media Player and other video players.

An option has been added to specify the same scaling for the X and Y axes. This is enabled with a checkbox on the **Plot Setup** screen. This is used for plots of trajectories of points on the vehicle and/or moving objects, such as offtracking plots.

When VS Visualizer is launched by clicking the **Plot** button on the **Run Control** screen, it skips animator assets to load more quickly. This helps diagnose runs that crashed, which sometimes cause the animator part of VS Visualizer to also crash. Also, the load time for VS files has been reduced dramatically, especially for larger files.

When VS Visualizer is showing plots with overlays from two or more output files, dragging new variables from the Data Manager window into a plot window adds the variables from all open files that contain those variables.

The default plot format was changed to match the style set in the **Plot Format** datasets shipped in recent versions of CarSim, providing a more consistence appearance covering plots set up from the database and plots created interactively.

Animation Resources

The amount of animator resources provided in CarSim has been greatly expanded. All current vehicle shapes for CarSim, TruckSim, and BikeSim are now provided in all three products, in support of ADAS examples with a variety of traffic vehicles. The `Resources/Animator` folder in `CarSim_Prog` has animator assets organized into four top-level folders:

1. `2D_HUD_Icons`
2. `3D_Shape_Files`
3. `Audio_All_VS`
4. `Road_Materials`
5. `Depreciated Assets`

Older versions of the software had many example shapes that used older formats as required by the obsolete SurfAnim program. Shape files have been upgraded to take advantage of more modern options, such as compression and lighting/shadow support.

The folder `Deprecated_Assets` contains assets that were used in 2017 but are no longer used. It is provided support of databases imported from older versions.

This release has other new features, such as alternative drivers (male and female, Caucasian and Asian) and more traffic signs.

New Tools

Version 2018 introduces new interactive tools for assembling scenes, paths, and road surfaces for ADAS examples. These include:

- **VS Scene Builder** uses drag-and-drop tiles to rapidly assemble scenes with 3D animation assets. The tool also provides click and drag options for creating VS reference paths to use with CarSim close-loop controllers and moving objects. Access this tool from the **Tools** menu of CarSim.

- The Atlas web server (atlas.carsim.com) introduced in CarSim 2016 was extended to the third generation to support the new import tool and to provide elevation data associated with downloaded paths.
- A new plugin for the ADAS RP application leverages the HERE map database for detailed road information with an option to query elevation data using Google via the CarSim Atlas web service. It combines horizontal coordinates from HERE with Z coordinates from Google to convert real-world map information into path and road datasets that can be used within CarSim.

These new tools generate JSON (JavaScript Object Notation) text files that can be imported to CarSim with the new **Scene: External Import** screen described below.

Graphic User Interface (GUI)

Scene: External Import Screen

A new library **Scene: External Import** was added to support import of path, road, and animation datasets from Atlas, ADAS RP, and VS Scene Builder. Datasets from this library are linked to datasets in the **Road: 3D Surface** library.

The imported datasets are unique in having both path/road data for the VS Solver, along with supporting 3D animator assets.

Preferences Screen

The **Preferences** screen was modified to fix some existing problems and support new features such as the increased number of animation resources. Past versions of the screen included a few blue links for default datasets to be used for Plot Formats and for shapes used to show targets in ADAS applications. These links led to problems in some cases when transferring data with CPAR files, so they were removed for version 2018. Here are some changes that were made:

1. A checkbox was added to support a new option: “Perform batch run updating when deleting or renaming Run screen.” This is helpful for users who make batch runs with the **Batch Runs** tool. When this is active and a dataset from the **Run Control** screen is deleted or renamed, the Browser scans all datasets in the **Batch Runs** library and fixes invalid references to the old name
2. A **Secondary Resources** field was added for specifying a location for VS Visualizer to find additional assets. Users can now add a folder with custom assets.
3. Checkboxes for selecting the deprecated tools WinEP and SurfAnim were removed; CarSim 2018 does not support those tools.
4. The blue link for a default plot format has been removed. New versions have links to a plot format for every **Plot Settings** dataset. Imported plot datasets without a specified plot format will use the default from VS Visualizer.

5. The blue links for default cube and cylinder used for ADAS-related animations have been removed. The link to 1x1x1 cube or cylinder is now hard-coded. (It still applies only when a box is checked to view the pure target shape.)

Moving Object Screens

The **Multiple Moving Objects** screen was changed to support new features. A wheelbase parameter and option were added in support of offtracking. The VS Commands that are written automatically make use of the new speed variable `V_OBJ_`*o*. `EQ_DIFFERENTIAL` commands that were used in past versions are no longer needed and are no longer written to the Parsfiles.

When the checkbox **Show target?** is checked, a rectangular or cylindrical shape is shown by VS Visualizer if a simulation using the dataset is viewed. In past versions, the 3D shape file was set in the **Preferences** screen. As noted in the previous subsection, the database now automatically uses a 1x1x1 cube or cylinder shape.

The **Single Moving Object (Custom)** screen was also changed to support the new speed variables and remove some automatically written VS Command equations.

Miscellaneous Changes

Here are some other changes to the GUI.

- Many of the miscellaneous yellow fields are now resizable to provide a quick view of the contents without scrolling.
- The **Tools** menu adds support for MATLAB 2017b.
- The **Plot Setup** screen was modified to remove a checkbox that only applied for the obsolete WinEP plotter, and to add support for the 1:1 X-Y scaling that is now supported by VS Visualizer.
- The **Powertrain: Shift Schedule** screen has two new fields for the upshift and downshift duration (keywords: `T_SHIFT_UP` and `T_SHIFT_DOWN`). If the new fields are blank, the default duration (legacy) from the **Powertrain: Transmission (18 Gears or CVT)** screen is used.
- Yellow fields were added to the **Brake System** screens to support the new bearing friction parameters (keyword = `MY_FRICTION`).

Database

New Examples

Examples introduced in versions 2017.0 and 2017.1 have been moved into other categories. Most examples that were introduced in 2018 have categories whose names begin with "`* CS 2018`".

1. CS 2018 – Camera Sensor: Highway Lane Detection (Simulink). This example shows the Camera Sensor capability of VS Visualizer when used with the Computer Vision System Toolbox and Image Processing Toolbox from MathWorks.

2. CS 2018 – Column Assist Power Steering. Examples show different options for using the new column assist power steering features.
3. CS 2018 – New Animator Assets. A few runs were made that show some of the new animator assets added to CarSim.
4. CS 2018 – Scene Import. New datasets exercising the new Path import method show scenes, paths, and roads imported from ADAS RP, VS Scene Builder, and Atlas.
5. CS 2018 – Sensor Improvements. Some examples show cases where better occlusion behavior of ADAS sensors is obtained.
6. CS 2018 – Twist beam 2018. The new suspension 2D kinematics and compliance tables are used to reproduce K&C data for a twist-beam suspension.
7. CS 2018 – Variable Effective Rolling Radius. This example shows the effective rolling radius of each tire that is not constant as specified on the Tires screen, but it varies in time as imported from Simulink through the new import variables `IMP_RRE_L1`, ...
8. CS 2018 – Various Shift Duration. This example shows the different shift duration time that can now be specified separately for upshifting and downshifting at each gear position.
9. Validation of Vehicle Data – examples were added with simple tests that are helpful for evaluating new vehicle datasets, such as a vehicle at rest on a flat level surface. These are described in a new Tech Memo “Validation of VS Vehicle Models.”

Changes Made to Existing Examples

Nearly all links to animator resources were changed to work with the new organization of animator resources mentioned earlier.

Animator sound sets were removed for most example. Although audio is an essential feature of CarSim when used in driving simulators, it is more of a distraction when using the tool in other applications, especially when the sound samples are not customized for the specified vehicles. An example run in the category “VS Visualizer Examples” still shows how audio can be used for engine, wind, and tire sounds as part of the video generated by VS Visualizer.

Documentation Updates

Documents from the Help Menu

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

1. ADAS RP Plug-In (Tech Memo)
2. Twist Beam Model with 2D Tables (Tech Memo)
3. Scene External Import (Screen Documentation)
4. The Design Load Condition in CarSim and TruckSim (Tech Memo)
5. Validation of VS Vehicle Models (Tech Memo)

6. VS Scene Builder (Tools submenu)
7. 3DS Max OBS OSG Export Setup (Tools submenu)
8. VS Output/Table API (Reference Manual)

The following Reference Manuals have been updated:

9. System Parameters in VS Solvers
10. VS API
11. VS COM Interface
12. VS Commands
13. VS Solver Programs
14. VS Visualizer

The following Screen documents have been updated:

15. ADAS Sensors and Moving Objects
16. Animator: Camera Setup
17. Animator Shapes and Groups
18. Animator Sounds
19. Animator: Wheel Arrows and Other Indicators
20. Custom Forces and Motion Sensors
21. External Models and RT Systems
22. Plot Screens
23. Powertrain System
24. Preferences
25. Run Control Screen (Home)
26. Setting Up Import and Output Variables
27. Tire Models
28. Vehicle Screens and Outputs

The following technical memos have been updated:

29. Example: Camera Sensors
30. Example: Running Multiple Vehicles in Simulink
31. Making Advanced Procedures with VS Commands
32. Numerical Integration in VS Solvers
33. Vehicle Dynamics Terminology

The following documents involving RT systems and driving simulators have been updated:

34. Concurrent RT Guide
35. dSPACE RT Guide
36. LabCar RTPC Guide
37. NI RT Guide
38. RT Lab Guide

Machine-generated documents

The machine-generated documentation for output variables includes more details for some variables, such as coordinate systems, details of definitions, etc. If the name of the variable is more than 8 characters in length, a shorter name is listed that can also be used.

The machine-generated documentation for state variables now include ODE state variables that were created with the `EQ_DIFFERENTIAL` command.

Values of state variables created with the `EQ_DIFFERENTIAL` command are now listed in the End file, in support of continuation runs.

The math models within the VS Solvers include some variables that were introduced in machine-generated code as independent state variables that are then constrained to be algebraic functions of other state variables. Past versions listed the constrained state variables in machine-generated documentation (accessed with the **View** button in the lower-right corner of the **Run Control** screen), with the misleading implication that they could be given initial condition values. In version 2018.0 they not listed.

Bug Fixes and Errata

Miscellaneous

1. The occlusion of moving objects that are detected by ADAS sensors was based on sorting the objects based on distance from the sensor. The simple method used in past version did not always work as intended in some cases where targets have drastically different sizes.

2. Tire forces and moments at the center of tire contact (CTC) may be imported. However, if the selected tire model was an external model such as TASS MF-Tyre or COSIN/Ftire, the import variables were not used in the multibody model, but were added to the force and moment output variables. This was misleading. In version 2018, these imports are ignored completely for external tire models (`OPT_TIRE_MODEL` ≥ 8) and do not produce confusing outputs.

3. Screens for tabular tire data have an option for **Legacy Tire (2D, absolute slip, from zero)**. When selected, the tabular data should use only positive values of slip. However, if asymmetric data were entered that started with negative slip angles, there was no error reporting. This has been fixed. As a result, there may be cases where datasets from previous versions will now generate errors.

4. Texture render issues which occurred when viewing Mcity scenes in VS Visualizer's compatible mode have been corrected.

5. The api-ms-win-crt-runtime*.dll error when using Windows 7 SP1 has been addressed with v2018.0.

6. Run and CreateAllParfile COM functions failure on locked screen has been fixed. User may have to unlock a screen before executing COM functions which would modify screen data.

7. Compatibility problems with MATLAB 2017a and 2017b have been corrected.

Backward Compatibility

Most of the new features involve new parameters and output variables. Old datasets do not use these features, so backward compatibility is maintained. However, some of the new features replace older capabilities, making some differences inevitable.

CarSim 2018.0 will automatically convert databases going back to 7.0 (2007). If you need to work with pre-2007 datasets (version 6), import into a CarSim version between 7.0 and 8.1.1 and then import from there into CarSim 2018.

Output variables for optional parts

As mentioned earlier, the default behavior for VS Commands that create new output variables was changed in v2018 to define the variables, but not activate them for writing to file. Instead, they are activated as needed for use in animations and plots (the same as built-in output variables).

We have found a few cases where output variables for optional parts (e.g., reference points) are referenced in old databases for use in VS Visualizer for plotting or animation before the variables are created. Because the variables were always activated for writing, plots and animations worked as expected. However, with the new behavior, the variables are not activated because they do not exist when the relevant Parsfiles are read.

A quick workaround is to activate all output variables for writing to file using the checkbox on the **Run Control** screen. However, this can generate unnecessarily large files, especially when the simulation involves many optional parts such as moving objects and ADAS sensors. The recommended fix (used in all v2018 examples) is to make sure links to datasets that define the new parts (reference points, motion sensors, etc.) are processed before links to datasets that use the new output variables in plots or custom animations.

Miscellaneous

1. Simfiles must have a vehicle code in order for the VS Solver to activate the appropriate modules. This is done automatically when runs are made from CarSim GUI. If running simulations from external software, be sure the Simfile includes the `VEHICLE_CODE` keyword and code.

2. New dynamic state variables were added for hitch deflections (X-Y-Z directions). Deflections are typically about 0.5 mm. Although agreement between 2018.0 models and

2017.1 is close (typically the first three or four significant digits), it is not exact. For more information about these effects please see the memo *Modular Vehicle Models in Version 2018*, available from the Help menu: **Help>Release Notes>CarSim 2018: Modular Vehicle Models**.

3. More error handling has been added in recent versions. When using datasets imported from older versions, especially those prior to version 9, errors might be reported. In most cases, the errors occurred in the older versions but were not noticed.

4. Variables that were listed as state variables but are not truly independent were removed from the lists of state variables. For example, road steer variables (`SV_STR_L1`, `SV_STR_L2`, etc.) are no longer listed. References to these variables are no longer valid. For example, the steering variables were specified for the linearization examples in 2017.1; they were removed from the linearization examples provided in v2018.

5. Output variables used to support fake shadows for animations in the obsolete SurfAnim program have been removed from the solvers.

6. One of the output variables installed by the VS Command `INSTALL_DRIVER_PREVIEW_SENSORS` was renamed for consistency with other outputs. The old name was `X_RdS_` *i*; the new name is `X_Targ`*i*. Both names are installed and may be used to activate outputs for writing/export. Both may also be used in VS Command formulas. However, only the new name can be used to define plots for VS Visualizer.

> **Note** The old name `Y_RdS_` *i* was intended to be renamed as `Y_Targ`*i*. However, there was a mistake that was not noticed until the last minute. Therefore, we have a new name for the X coordinate but an old name for the Y coordinate.

7. Examples provided in past versions for running the VS Solvers from custom "wrapper programs" used the old convention of having a separate solver for each vehicle configuration (e.g., `s_s.dll`, `s_ss.dll`, etc.) located in the folder `Programs\solvers\Default` for 32-bit versions and `Programs\solvers\Default64` for 64-bit versions. Those old wrapper programs will not work with the new convention of having just two modular solvers: `Programs\solvers\carsim_32.dll` and `Programs\solvers\carsim_64.dll`. Any custom wrapper code that used the old name and locations of the VS Solver DLL will need to be modified, as is done with the example source code provided in v2018.

8. The default behavior for VS Commands that create new output variables was changed in v2018 to define the variables, but not activate them for writing to file. (This is the same behavior as all built-in output variables.) Variables are activated for writing by using them in datasets from the **Plot Setup**, **Animator Reference Frame**, and **I/O Channel Output** libraries.

Be aware that output variables cannot be activated until after they are defined. Therefore, datasets to create new parts (sensors, moving objects, etc.) must be referenced before the datasets that will make use of them.

9. The solver now checks for exceptional floating-point values such as NaN (not a number) at simulation time. When importing a database from an earlier version of CarSim, some existing VS commands equations may need to be rewritten. For example, the VS Commands equations for engine sounds in the 2017.1 database should be changed to use the ABS function before applying the SQRT function, to avoid producing a NaN value.

Known Issues

1. When working in Windows 10 with multiple monitors that have different DPI settings, the menus of CarSim 2018 (and other recent versions) do not position correctly on at least one of the monitors. Once a CarSim menu is activated, the mouse position is offset or wrong until the menu is dismissed or an item is selected. To avoid this behavior, ensure that the CarSim pop-up windows are positioned on the screen where you intend to work.

2. When using dSPACE 1006 and 1005 HIL boards, output names greater than 8 characters will not be received by VS Visualizer during live animation. Be sure to use short names on these platforms.