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

CarSim 2018.1 New Features

The CarSim Installer 2
VS Solver: Architecture 2
Recommended Option for Numerical Integration 2
Initialization 3
Connection Timing for Import and Export Variables 3
Timing for VS Commands and API Function Calls 4
Simulink S-Functions 4
Embedded Python for Windows 5
PyFMI 5
Miscellaneous 5
VS Solver: Models 6
ADAS Sensors and Targets 6
Improved Management of Configurable Functions with Many Datasets 7
Miscellaneous 7
VS Visualizer 8
VS Scene Builder 8
VS Connect 9
VS Browser: Graphic User Interface (GUI) 9
Model Screens 9
Moving Object Screens 9
Run Control and Procedures Screens 10
Events Screen 10
Speed Control (Closed Loop) Using Path Preview 11
Preferences Screen 11
Screens with User IDs 11
COM Automation Error Handling 11
Road: Animator Surface Shapes Screen 11
Scene External Import 12
Path: X-Y Screens 12
Documentation 12
Database 14
Hybrid Powertrain Model 14
New and Updated Examples 14
Updated CPAR Archives 15
Animator Assets 15

This document lists the notable new features in CarSim version 2018.1, organized into sections based on the main components of the software package.

In previous releases, a single New Features document described new features, bug fixes, and information regarding backward compatibility with older databases. Starting with this version, a separate document is included in the **Help** > **Release Notes** submenu: CarSim 2018.1: Backward

Compatibility. This document includes information about bug fixes and backward compatibility issues for recent releases going back to version 9.0 (released in 2014).

Going forward, we plan to update this backward compatibility document by including information about the new release (e.g., 2019.0). Content involving previous releases will remain in the document and serve as an archive of past issues and associated fixes, assisting end-users who are still working in older versions (e.g., version 2016.2) but may not upgrade sequentially — i.e., 2016.2 to 2017.1 to 2018.1.

The CarSim Installer

After the initial installation of CarSim, the installer `Setup_CarSim_2018.1_r`*rev*`.exe` (where *rev* is a revision number) may later be used multiple times to create new, clean `CarSim_Data` folders. When used to install a new `CarSim_Data` folder, the installer for 2018.1 is a little friendlier to use than in past releases: if a folder is selected that is not empty, the installer will make a new one with the same root name, using the Windows convention of adding a suffix to indicate the name was already used. For example, if an existing folder named `CarSim_Data` is selected, the installer will create a new one named `CarSim_Data (1)`.

A warning about needing administrator permission was changed to clarify that such permissions are not needed to install a new database folder.

VS Solver: Architecture

Recommended Option for Numerical Integration

In past versions of CarSim, the recommended numerical integration method was AM-2 (Adams-Moulton 2nd order), which calculates model equations twice per time step. With AM-2, the typical time step (`TSTEP`) is 0.001s; this means that at the half-step, the model equations are calculated at intervals of 0.0005s.

In reviewing typical uses with external software and user-defined extensions from VS Commands, Mechanical Simulation now recommends two options for different situations. The recommended integration methods are as follows. All existing simulations in the database have been updated to follow these guidelines.

1. AB-2 (Adams-Bashforth) method as the default, with `TSTEP` set to 0.0005s.
2. AM-2 with `OPT_IO_UPDATE` = 0 when using external models that do not need rapid communication, or which are required to run at 0.001s (a common standard for controller models). Note that when `TSTEP` is set to 0.001s, the VS Solver calculates variables internally every 0.0005s.

For more information, please see the technical memo *Numerical Integration in VS Solvers*, which has been updated for this release.

Initialization

Internal keyword database

The creation of the internal database of keywords (parameters, commands, output variables, state variables, etc.) was reworked to handle the very large number of keywords installed with ADAS scenarios with many moving objects (up to 200) and range and tracking sensors (up to 99). With the maximum number of detection variables (99 sensors x 200 targets x 22 variables/detection resulting in 435,600 output variables), the initialization time was reduced from several minutes to 6s.

Model initialization

The initialization process has been modified such that built-in equations and VS Commands are applied twice at the start of the simulation (`T = TSTART`) when `OPT_SKIP_INIT_DYN` is zero. This has two benefits:

1. Compliance effects due to vertical load are calculated at the first time-step, eliminating some transient behavior seen in past versions.
2. When working with external software that connects to CarSim with Import and Export arrays, imported variables are applied at the very start (as described in more detail in the next subsection).

Connection Timing for Import and Export Variables

The communication between a VS Solver and an external simulation workspace (e.g., Simulink, FMI) has been improved.

Initialization

In past versions, the model equations were applied at the start time (`T = TSTART`) to calculate all state variables and output variables before writing the Echo file. Imports from external models were not yet available for initialization. As a workaround, import variables had associated numerical values that were used for initialization purposes. The first time the VS Solver was contacted by the external workspace, export variables were provided as calculated by the solver, but import variables were not available until the next time step (`TSTART + TSTEP`).

The initialization process has been reworked such that the calculations of all model variables are made after the import variables have been obtained. The first time the VS Solver is contacted by the external workspace, import variables are available. The availability of variables in the export array is controlled by a new system parameter `OPT_SKIP_INIT_DYN`:

1. If `OPT_SKIP_INIT_DYN` is zero (the default), the model equations are applied before the Echo file is written. The Echo file will include the calculated location of the total vehicle unit(s) mass CGs and list initial values of any exports. The first time the VS Solver is contacted by external software, the exported variables will have calculated values.
2. If `OPT_SKIP_INIT_DYN` is nonzero, the Echo file will not have the calculated location of the total vehicle unit(s) mass CGs. The first time the VS Solver is contacted by

external software the export values will all be zero. This option is provided in case external equations are not valid if applied twice at the same time (`T = TSTART`).

Synchronized import and export variables

When running under the control of external software (Simulink, LabVIEW, FMI, etc.), the VS Solver appears as a block with arrays of import and export variables. Traditionally, all equations in the VS Solver block are applied in a fixed sequence for time T, which is specified by the external software. This works perfectly if the variables imported into the VS Solver block are based on factors external to the vehicle model, such as clock time. However, another type of connection occurs when parts of the multibody model are defined externally, such as tires, springs, powertrain, etc. Deflections and speeds exported from the VS model are used to calculate forces and moments that are provided as imports to the VS model at the next time step. In this case, the imported variables have a time lag.

An alternative timing sequence is now available in which kinematical variables (positions, velocities, deflections, rates) apply for time T + ΔT, while all other variables (forces, accelerations, miscellaneous outputs) still have values calculated for time T. At the next time step, forces and moments imported from the external model are synchronized with the new time T + ΔT.

A new system parameter `OPT_IO_SYNC_FM` has been added that is not visible unless the numerical integration method is Euler or AB-2 (these are set with `OPT_INT_METHOD` set to –1 or 0, respectively) and the simulation includes both import and export variables. When this parameter is nonzero, forces and moments calculated externally are sychronized with the rest of the vehicle multibody model.

Timing for VS Commands and API Function Calls

In past versions, calculations could be added at the beginning of a time step with the VS Command `EQ_IN`, or at the end of a time step with the VS Command `EQ_OUT`. A new location in the time step with a corresponding VS Command, `EQ_DYN`, has been created. Equations added with this command are applied after the model kinematical variables have been calculated, but before the built-in forces and moments are calculated. This allows user-defined equations to make use of current kinematical variables to calculate forces and moments for immediate use in the time step.

A corresponding command and location has been added for calculations made using the VS API: `VS_EXT_EQ_DYN`.

Simulink S-Functions

Prior to version 9.0, the S-Function used to connect the VS Solver to Simulink supported the automatic creation of ports to handle detections of moving object targets by ADAS sensors, with ten detection variables. With version 9.0, a second-generation S-Function was introduced that had better memory management and supported multiple ports. The second-generation S-Function icon that appeared in Simulink was identified by a “2”.

The capabilities of the two S-Functions have been merged: the new VS S-Function (`vs_sf`) supports both the automatic port-creation of the original S-Function, plus the improved memory

management and user-defined ports from the second-generation S-Function. For CarSim 2018.1, the new S-Function has the text “vs_sf” in the icon.

Embedded Python for Windows

Python 3.6.5 has been embedded in the VS Solvers and can be accessed with three new commands.

1. The VS call `RUN_PYTHON_STRING` executes a text string as a python command line.
2. The VS call `RUN_PYTHON_PROG` executes a python function and can accept input and output in the form of VS step tables.
3. The function `PYTHON()` can be used in equations to call the embedded Python interpreter, typically at each simulation step. The new function can also accept VS data values as I/O.

This feature is supported for Windows VS Solvers; it is not available for Linux or RT systems.

PyFMI

Previous versions of CarSim supported the ability to generate functional mockup units (FMUs) in co-simulation mode. This version supports the ability to run these FMUs with Python along with PyFMI, if Python 2.7.x is the installed on the computer.

Miscellaneous

1. New system constants were added and are visible in the Echo file: `DR`, `G`, `PI`, and `ZERO`. The constants `DR`, `G`, and `PI` existed in past versions but were not listed in the Echo file. The constant `ZERO` is provided to simplify the use of VS Events when evaluating formulas as being either zero or not zero.
2. A new system parameter was added: `OPT_ECHO_KEYWORDS_LC`. When set to a nonzero value, all keywords and commands listed in the Echo file are written in lowercase. This is helpful when copying equations or keywords to paste into Miscellaneous yellow fields, given that lowercase words take up less space.
3. Table entries of type `STEP` can now be updated after definition or loading with an equation. (e.g., `eq_init example_tab(0,1,1) = 10`).
4. Attempts to set units of Configurable Functions with the VS Commands `SET_UNITS_TABLE`, `SET_UNITS_TABLE_COL`, and `SET_UNITS_TABLE_ROW` would perform no action if the keywords for the table function or units were not recognized. These now generate errors telling which keyword was not valid.

VS Solver: Models

ADAS Sensors and Targets

New speed control options

New parameters were added to simplify the use of moving objects to represent traffic vehicles and pedestrians. If the object is set to follow a path (specified with the parameter PATH_ID_OBJ), then several options are built-in for specifying speed with the parameter OPT_SPEED_OBJ. If one of these options is selected, then the variable V_Obj_*o* (where *o* is the object number) is calculated, and station along path S_Obj_*o* is automatically upgraded to be a state variable handled with an ordinary differential equation (ODE).

A convenient means for specifying V_Obj_*o* is with the SPEED_TARGET Configurable Function. A new parameter for each object, SPEED_ID_OBJ, has been added to simplify the use of the SPEED_TARGET function. If SPEED_ID_OBJ is nonzero, then:

V_Obj_*o* = SPEED_TARGET(S_Obj_*o*, T, SPEED_ID_OBJ) (SPEED_ID_OBJ > 0)

V_Obj_*o* = -SPEED_TARGET(S_Obj_*o*, T, ABS(SPEED_ID_OBJ)) (SPEED_ID_OBJ < 0)

If an object is tied to a road surface via the parameter ROAD_ID_OBJ, then the elevation Z_OBJ_*o* of a moving object and the pitch and roll angles (PitchO*o* and RollO_*o*, respectively) are calculated automatically using the road surface geometry at the location of the object. This approach is well suited for orienting traffic vehicles to match road grade and cross-slope. However, it is not ideal for pedestrians and bikes. New parameters have been added to allow the pitch and roll to be calculated independently of the road surface (these are OPT_ROAD_PITCH and OPT_ROAD_ROLL).

New pass-through variables

Moving objects have two new output variables: TypeO_*o* and MsgObj_*o* (where *o* is the object number). Both are nominally zero and will remain that way unless the value is set in a Miscellaneous yellow field or imported. If the object is detected by a sensor, these values are passed through to corresponding detection variables TypS*s*_*o* and MsgS*s*_*o* (where *s* is the sensor number and *o* is the detection number).

These custom outputs are used in some new examples: for speed limit signs, the type indicates that the object is a speed limit sign and the message provides the speed limit; for a stop sign, the type indicates the object is a stop sign (or red light) and the message identifies the distance from the sign to the target where the vehicle should stop.

More support for outputs handled as parameters

Depending on the setup, the station variable S_Obj_*o* might be:

1. A constant (e.g., location of a stop sign).
2. Updated each time-step using VS Commands (e.g., a traffic vehicle some distance in front of the ego vehicle).

3. An ODE state variable, calculated by integrating the speed variable V_Obj_*o*.

Object variables that are active and might be constant have been installed as parameters, such that they are listed in the Echo file with other properties of the moving objects. These include the new TypeO_*o* and MsgObj_*o* variables. If an object is associated with a path, the variables S_Obj_*o*, LatO_*o*, HeadO_*o*, and V_Obj_*o* are also listed. If an object is not associated with a path, variables X_Obj_*o* and Y_Obj_*o* are listed instead.

Improved Management of Configurable Functions with Many Datasets

Some Configurable Functions support multiple datasets that are added as needed. Three of these were upgraded for this release:

1. LTARG (lateral target relative to a path), used by the built-in driver model and for traffic vehicles.
2. ROAD_DZ, used to add layers of vertical displacement to road surfaces for curbs, bumps, etc.
3. SPEED_TARGET, used for the built-in driver speed controller and for traffic vehicles.

Each of these functions now has a user ID that can be assigned an integer of 999 or greater. New commands are installed to support automatically searching for a specified ID number and either setting the context to the appropriate function or adding a new dataset if one doesn't already exist.

The default values of the number of LTARG datasets (the parameter N_LTARG) and SPEED_TARGET datasets (N_SPEED_TARGET) were changed from 1 to 0 to simplify the management of these datasets.

The GUI screens for the closed-loop path-follower and speed control were modified to specify whether data on the screen should either be added to the model or replace the content for an existing dataset. New VS Commands were added to support the management of Configurable Functions with User IDs.

Most of the User IDs were checked to ensure that they were either the same as the index (e.g., ROAD_DZ_ID = IROAD_DZ) or integers of 999 or greater. The error checking now allows the IDs to be set with symbolic formulas, including user-defined parameters.

Miscellaneous

1. Lateral and longitudinal compliance deflections for a solid axle are now available as output variables: CmpX_A*a* and CmpY_A*a*, where *a* is the axle number.
2. The parameter GPS_REF_ALT was added to provide the GPS altitude where Z = 0. This adds to the existing parameters GPS_REF_LAT and GPR_REF_LONG that define where X and Y = 0.
3. Added an output variable GPS_Altitude, defined as Zo + GPS_REF_ALT.
4. Changed the algorithm used to obtain curvature for VS Command functions PATH_CURV_ID and ROAD_CURV_ID to have less "noise" when working with spline table functions.

VS Visualizer

VS Visualizer is now available as a 64-bit version, supporting large simulations with binaries exceeding 4GB of data. The VS Visualizer platform can be changed between 32-bit and 64-bit from the CarSim **Preferences** screen, complementing the 32-bit version included with previous releases.

Additional keywords have been added to VS Visualizer to adjust the environment lighting. `CAM_GLOBAL_DIFFUSE` and `CAM_GLOBAL_SPECULAR` will adjust the lighting properties of the primary light source.

The Ambient Light setting has been removed from the VS Visualizer preferences and replaced with a Brightness Modifier. The Brightness Modifier is a numeric field where 1 indicates true light settings, less than 1 will make the scene darker, and greater than 1 will make the scene brighter. Any ambient light adjustments can be made from within the VS Browser using the keyword `CAM_GLOBAL_AMBIENT`. For examples, use the Find Text option (**Tools** > **Find Text in the Database)** and search for `CAM_GLOBAL_AMBIENT.`

The local time for an animated shape (walking pedestrian, bicycle, etc.) can be set using a new command `SHAPE_SET_TIME_FUNCTION,` followed by a formula for the local time. This is used in examples for walking pedestrians to synchronize the striding motions of the legs and feet to match the forward speed, such that there is no visible slipping of the feet on the ground surface.

VS Scene Builder

The VS Scene Builder has benefited from the addition of many scene props that can be dragged and dropped into the scene. Examples include animated objects, barriers, and signs.

Path selection and display have been improved. Hovering the mouse cursor over the path gives the path station and global X-Y coordinates at a specific point. Selecting a path from the path list will show a quick animation to help locate the path, and double-clicking on the path name will center the scene view around the extents of that path.

Group-dragging and automatic tile snapping based on compatible edge points have been vastly improved. The selection and snapping state are now clearly indicated using visual cues.

Right-click context menus are now available. This allows for setting the drawing order of scene objects, setting the zoom level of the view to encompass the entire scene, and setting a custom height for added props.

All commands that modify the scene can be removed by the Undo command, and Redo allows for those commands to be re-applied.

A Re-export option has been added to the File menu to increase iteration speed while using the Scene Builder. A complementary Re-import button has also been added the Scene: External Import screen in the VS Browser.

Additional bug fixes and improvements are described in the VS Scene Builder manual.

VS Connect

VS Connect is a new communication library intended to simplify data synchronization for co-simulation. The CarSim 2018.1 release includes:

1. VS Connect C library, for use in custom C programs
2. VS Connect S-Function, to facilitate communication with Simulink models.

In addition, the CarSim Plugin for Unreal Engine has been upgraded to support VS Connect communication. The plugin can be downloaded from the Unreal Marketplace: www.unrealengine.com/marketplace/carsim-vehicle-dynamics

The example project on the Unreal Marketplace can be used in conjunction with the new MATLAB/Simulink-based CarSim dataset to demonstrate co-simulation using Unreal Engine and Simulink.

VS Browser: Graphic User Interface (GUI)

Model Screens

CarSim includes 12 Model libraries (**Models: Simulink**, **Models: Self-Contained Solvers**, etc.). All include options for setting the numerical integration method and time step size, and all have been updated to offer the same options. If the boxes are checked to set the integration method, time step size, and possibly the synchronization of the export variables with the import variables concerning forces, then the intention is for these settings to override what is linked on the **Preferences** dataset. If not checked, settings from the most recently viewed **Preferences** dataset are used for the run.

Moving Object Screens

Changes were made on two screens used to add moving objects to a simulation. Both involve the visualization of the object and/or sensor target.

Blue link(s) to animator dataset(s)

The **Single Moving Object (Custom)** screen includes a blue link to an animator dataset associated with the object. In past versions, the library was fixed to the library **Animator: Vehicle and Sensor Targets**, which is also used to provide animation information for the sprung masses of simulated vehicles. The link has been extended to allow other animation libraries to be used, such as **Animator: Shape File Link** and **Animator: Group**.

The **Multiple Moving Objects** screen supports up to 12 new object definitions, each with its own blue link to an animator dataset. In this case, the blue link for the first object is extended to allow other animation libraries to be used. Whatever library is chosen for the first object is also used for the blue links for animation datasets for the other objects.

Display of color selector

Each screen also includes a color selector that was shown when a box **Set color?** was checked. The color selector is now shown if either of two boxes are checked: **Set color?** or **Show target?** If neither are checked, the color selector is hidden.

Run Control and Procedures Screens

More options for Plot Setups

The **Run Control** and **Procedures** screens include lists of **Plot: Setup** blue links that were fixed to the **Plot: Setup** library in past versions. Both have been modified to allow groups of **Plot: Setup** datasets to be specified.

Two of the links on the **Procedures** screen have been extended to allow generic libraries to be used. Datasets in the Generic libraries can in turn be linked to multiple **Plot: Setup** datasets. For example, the **Generic VS Commands** screen has 16 links that can be used for **Plot: Setup** datasets. On the **Procedures** screen, the first and tenth links have been extended. Whatever library is used for the first link will also be used for the next nine links. Similarly, whatever library is used for the 11th link will also be used for the next eight.

The **Run Control** screen has less available space, so only one of the links is extended: Link 6. Links 7 and 8 are set to whatever library is used for Link 6. Links 1-5 remain fixed to the **Plot: Setup** library.

Improved management for speed control

The screens Target speed vs. time/station and Target speed vs. station have settings to specify a closed-loop speed controller. The Configurable Function for target speed includes up to 200 datasets, in support of ADAS scenarios where target speeds can be assigned to moving objects or used by the ego vehicle during different parts of a complicated scenario.

In past versions, the target speed specified on either screen applied to the first one (`ISPEED` = 1), possibly overriding an existing dataset. In both screens, selecting a closed-loop speed control option now has the effect of adding a new dataset (`N_SPEED_TARGET` is incremented) and setting the ego vehicle control (`SPEED_ID_SC`) to the new dataset. Existing datasets are left intact.

Events Screen

The **Events** screen was modified to include two new options for the simple cases of “If a formula is not zero” and “If this formula is zero”.

1. A new system parameter, ZERO, was added to the VS Solvers. This is the variable to monitor, with a ~= operator, and a symbolic expression that will trigger the Event when not zero.
2. Two more options were added on the Event drop-down control: If, and If Not. These use the existing ~= and ==.

The yellow fields for formulas are slightly larger, and now include resize controls on the right edge so they can be stretched to view longer formulas.

The option for setting a constant speed with the built-in speed controller has been extended considering the user ID now available for SPEED_TARGET datasets. An option for setting a custom ID is provided which allows the same dataset to be reloaded multiple times instead of making new speed datasets for every iteration.

Speed Control (Closed Loop) Using Path Preview

The option for setting a constant speed includes an option for setting a custom ID, allowing the same dataset to be reloaded multiple times instead of making new speed datasets for every iteration.

Preferences Screen

A new checkbox control was added to the screen to disable the writing of timestamp information in exported Parsfiles. This was done because new timestamps can cause confusion with some version control software.

A new drop-down control was provided to choose between 32/64-bit versions of VS Visualizer.

Right-click information for controls was updated to better describe the behavior in 2018.1.

Screens with User IDs

In 2018.0, most of the User IDs were checked to ensure that they were integer values set to 999 or larger. The error checking has been updated to allow the IDs to be set with symbolic formulas such as user-defined parameters.

The controls involving the User ID were modified such that the yellow fields are larger in support of specifying the ID with a symbolic formula.

The **Steering by the Closed-Loop Driver Model** screen is represented by the LTARG Configurable Function, used for the ego vehicle, traffic vehicles, and other moving objects. The screen now has a checkbox to allow the closed-loop steering controls to be hidden, allowing LTARG datasets to be created without concern of conflicting with driver model settings.

COM Automation Error Handling

The VS COM interface has been expanded to include functions that can return error information. For example, a batch run may return an error message for a specific run that failed. These new functions end in _CheckError. A complete list of functions can be found in the VS COM Interface technical memo.

Road: Animator Surface Shapes Screen

Drop-down controls for **Material Type** on the **Road: Animator Surface Shapes** screen have been extended to use submenus, providing a more user-friendly view of the options. More materials (textures, painted markings, etc.) have been added, and the **Help** document for the screen includes references to all of the material types.

The column names and sizes were adjusted so the table fits on the screen without resizing. Right-click help was updated for the columns, and the maximum number of rows was increased from 25 to 50.

Scene External Import

Several improvements were made to this screen.

GPS Reference Point

A field was added for the reference GPS altitude (`GPS_REF_ALT`) for the ground point at the reference latitude and longitude. This is mainly used when importing GPS data from Atlas or ADAS RP.

A checkbox was added to generate the reference GPS values (`GPS_REF_ALT, GPS_REF_X, GPS_REF_Y`) automatically using the first point of the imported path, such that the X-Y-Z coordinates of that point are (0, 0, 0). If not checked, values provided by the user are used to perform all calculations of X, Y, and Z from imported data.

With the new options, multiple paths can be imported that will all convert data to a single global X-Y-Z coordinate system. A new Tech Memo, “Connecting 3D Roads from Atlas” describes the example highway intersection made from three connected road surfaces.

New Views of X-Y-Z data

A drop-down control was added to choose among views for the X-Y-Z data. The original X-Y plot is still available, and a new option to view Z vs. S has been added. Both plot options show the data after smoothing. A third option is to show a 4-column table with the raw (unfiltered) X-Y-Z data along with S. The table can be viewed in spreadsheet or field mode, as is the case with tabular data on other screens. The tabular data may be edited to remove outlier points or make other changes.

Path: X-Y Screens

Three Path screens have a button named **Import GPS Coordinates**. The screens now include the GPS smoothing options available on the **Scene External Import** screen.

Documentation

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

1. 3D Shape Files for VS Products (Animator)
2. CarSim 2018.1: Backward Compatibility (Release Notes)
3. CarSim Unreal Engine Plugin Example using VS Connect (Guides and Tutorials)
4. Chassis Torsional Flexibility (Tech Memo)
5. Connecting 3D Roads from Atlas (Paths, Road Surfaces, and Scenes)
6. Road Surface Visualization (Paths, Road Surfaces, and Scenes)

7. VS Connect S-function (Tech Memo)

The following tutorials have been updated:

8. CarSim Demo Tutorial
9. CarSim Quick Start Guide
10. Running a VS Math Model in Simulink
11. Silent Installation for VehicleSim Products

The following Reference Manuals have been updated:

12. System Parameters in VS Solvers
13. VS API
14. VS Browser
15. VS Commands
16. VS Solver Programs
17. VS Visualizer

The following Screen documents have been updated:

18. ADAS Sensors and Moving Objects
19. Custom Forces and Motion Sensors
20. Driver Controls
21. Engine Mounts
22. External Models and RT Systems
23. Paths and Road Surfaces
24. Powertrain Systems
25. Preferences
26. Procedures and Events
27. Run Control (Home)
28. Scene External Import
29. Setting Up Import and Output Variables

The following technical memos have been updated:

30. Example: Camera Sensors
31. Example: Multiple Ports in Simulink for Sensors
32. Example: Running Multiple Vehicles in Simulink
33. Modular Vehicle Models in Version 2018
34. Numerical Integration in VS Solvers

35. Running a VS Math Model in Simulink
36. The Design Load Condition in CarSim and TruckSim
37. VS Scene Builder
38. VS Scene Tile Creation

Database

Hybrid Powertrain Model

A hybrid powertrain Simulink model which was originally made by the University of Michigan and modified by Mechanical Simulation Corporation has been connected to CarSim. The model is very similar to a current in-production hybrid system in which the engine, generator, and electric motor are connected by a planetary gear set. The model calculates the torques, speeds, and power distribution, and involves the battery, electrical circuit, and power management controller that enables the calculation of fuel consumption and battery power level. Three example maneuvers are provided: US06 acceleration, EPA urban cycle, and EPA highway cycle.

New and Updated Examples

Some runs originally introduced in 2018.0 have been updated to demonstrate 2018.0 to 2018.1 feature improvements, and some new examples were added to existing categories. Those runs are found under "`* CS 2018.0`" categories with an "`*`" prepending the name of the new or changed run.

1. CS 2018 - Camera Sensors: highway lane detection and pedestrian detection
2. CS 2018 - Column Assist Power Steering: new example for Understeer Gradient
3. CS 2018 - New Animator Assets: animated pedestrians in city
4. CS 2018 - Scene Import: updated ADAS-RP example, three new examples for a 3D highway intersection made with GPS data from Atlas

Categories of new examples that were introduced in this release have names that begin with "`* CS 2018.1`":

1. CS 2018.1 - ADAS Improvements: new deer, road sign, pedestrian, and traffic signal state detection examples
2. CS 2018.1 - Embedded Python: steer controller and update path to avoid collision example
3. CS 2018.1 - Hybrid Powertrain: fuel consumption examples demonstrating EPA Urban and Highway US06 Acceleration tests
4. CS 2018.1 - Kinematical Preview: set of examples demonstrating various integration techniques with kinematical preview either enabled or disabled.

Updated CPAR Archives

CarSim 2018.1 includes nine importable parsfile archives (.CPAR or .PAR files). Seven were updated for 2018.1, and one new CPAR was added: `Chassis_Twist_CarSim.cpar`. These examples that are discussed in a new Technical Memo, Chassis Torsional Flexibility.

Animator Assets

Animated pedestrians

Several pedestrian shapes have been added in pairs: (1) a walking version and (2) a waiting version. Examples are included to show how to automatically switch between the two modes: when the speed of the object is zero, VS Visualizer shows a pedestrian standing still; when the speed is not zero, the pedestrian has fully animated walking motions.

The animation of the moving arms and legs are set to match the forward speed using the new command `SHAPE_SET_TIME_FUNCTION` in VS Visualizer.

Animated shapes were added for a pedaling bicyclist and three versions of a deer (standing, walking slowly, and running).

A dataset for every animator asset

Most of the animator assets (shape files, textures, audio files, etc.) are contained in the `CarSim_Prog` folder where they are accessible to all databases, even those imported from past versions. Starting with the 2018.0 release, animator assets from BikeSim, CarSim, and TruckSim are provided for all three products. These are contained in the folder `Resources\Animator`, and include the following folders:

1. `2D_HUD_Icons`
2. `3D_Shape_Files`
3. `Audio_All_VS`
4. `Deprecated_Assets`
5. `Road_Materials`

Although version 2018.0 includes the asset files for all three products, many were not used in existing datasets. Version 2018.1 adds datasets that link to most of the new assets. Some are in the databases as regular datasets; others have been organized into CPAR archives that can be imported. For example, one CPAR has European road signs, another has USA road signs.

Road paint

Graphics have been added for common patterns painted on roads: crosswalks, yellow and white lines, bicycle lane, etc. These are available with pull-down menus on the **Road: Animator Surface Shapes** screen. Some are used in new examples, and all are described in new documentation.

Lighting, Sky Boxes, Environmental Spheres

The environmental spheres (sky and land) have been updated to make use of improved lighting options in VS Visualizer. These do not affect the multibody simulations, but animated results shown in VS Visualizer have a better appearance.