Paths and Road Surfaces

BikeSim, CarSim, and TruckSim include 3D multibody models that simulate vehicle dynamics under driver/rider control on a 3D ground surface. The equations of motion for the physics math models are formulated using global X-Y-Z coordinates; all positions, velocities, and accelerations are available in global coordinates, and sometimes, local body-fixed coordinate systems.

In most simulations, the vehicle is controlled to follow paths of interest, and tires only contact the ground near those paths of interest. In the VehicleSim (VS) math model, the concept of a *road surface* is mainly a representation of the ground properties (geometry, friction, and rolling resistance).

Two options are available for defining the road surface: VS Roads and VS Terrain. VS Roads use a form that is well-defined where the vehicle tires are likely to travel, and sparse or nonexistent where the tires are unlikely to be. To do this, a VS Roads represents the road surface using a coordinate system based on a 2D *VS Reference Path* — a continuous line that exists in a horizontal plane with continuity in position and gradient. This document covers VS Roads and the GUI screens used to create them.

VS Terrain is the second option, and is usually used for 3D data from external sources that were converted to the `vsterrain` file format, accessed with the VS Terrain interface. It is well suited for driving simulators and simulations involving high-quality visual rendering using graphics engines such as Epic Unreal. The VS Terrain option is described in the *VS Terrain* and *VS Scene Builder* tech memos and the *Scene External Import* screen document.

In addition to providing the coordinate system for describing VS Road properties, VS Reference Paths are used as targets for the driver model for steering and speed controllers, and to define motions of objects such as traffic vehicles. Managing some of these options involves extending the model beyond core vehicle dynamics and requires some thought and planning.

To accommodate these scenarios with greater complexity, VehicleSim models support up to 500 Reference Paths and up to 200 VS Roads. The road surfaces can be connected to efficiently describe the geometric and friction interface to the tires wherever the vehicle is likely to go. This allows the vehicle and other moving objects to use multiple road surfaces, which can be useful for simulations involving a change in the critically-important interface between the tires and ground.

If the simulation has just one VS Road and mainly involves the evaluation of the vehicle behavior, it is usually not necessary to assemble an environment with multiple road surfaces; use existing road datasets or make copies and adjust the properties to suit the needs of the test. This approach works well for users who plan to simulate vehicles in pre-defined scenarios and conditions.

Besides representing roads in the vehicle math model, BikeSim, CarSim, and TruckSim generate 3D shapes to visualize VS Road surfaces. The screens used to assemble the road shapes are described in the document *Road Surface Visualization* (**Help** > **Paths, Road Surfaces, and Scenes**).

Reference Paths

A VS Reference Path is a continuous line that exists in a horizontal plane with continuity in position and gradient. That is, there are no sharp corners.

The purpose of a Reference Path is to define a 2D coordinate system for describing locations near a path of interest. The path coordinates are station S (distance along the path) and a lateral coordinate L (distance a point is from the path, measured on a line that intersects the point and the path, and is perpendicular to the path at the point of intersection, Figure 1). In this coordinate system, the S coordinate is based on an axis (the path) that is fixed, but not necessarily straight, while the L coordinate is based on an axis whose location is variable (it starts from the S location on the path), and whose direction is variable (perpendicular to the path at S).

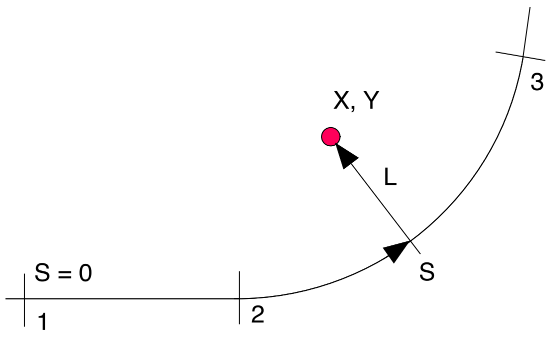


Figure 1. S and L are the coordinates of a Reference Path.

The expectation is that the S coordinate will cover a range of hundreds or thousands of meters, while the L coordinate typically covers a more limited range such as the width of a road.

Note In the special case where the path follows the global X axis, S=X and L=Y. In this way, the S-L convention reverts to the global X-Y coordinate system for applications where a curved path is not needed.

The main applications in VS Math Models for the S and L coordinates associated with a path are:

1. Closed-Loop steering controllers, which work either by minimizing the absolute value of the L coordinates of a preview point ahead of the vehicle relative to a target path or by aiming the front wheels toward a preview point which is along the path.
2. Objects (traffic vehicles, targets for on-board ADAS sensors, etc.) located relative to the path using S and L.
3. Lanes defined using constant values of L for a Reference Path, or possibly with the Configurable Function `LTARG` that defines L as a function of S.
4. Road surface elevation and friction defined as functions of S and L for a road Reference Path.

A VS Reference Path is a sequence of contiguous segments. For example, the path shown in Figure 1 might be defined with three segments: a straight line going from point 1 to point 2, a circular arc going from point 2 to point 3, and a straight line starting at point 3.

ID Numbers for Paths and Roads

There are three kinds of ID numbers that are used when dealing with paths and roads.

Internal Index Numbers

VS Solvers use an indexing convention for many parameters and Configurable Functions associated with parts that are added by users, such as payloads, sensors, and moving objects. Index numbers are automatically set to start with 1 for the first instance and are incremented for each new instance as datasets of the same type are scanned. For example, `SPATH_START(2)` is a parameter for the second path defined in a simulation; the internal index is 2.

Custom ID Numbers

Paths, road surfaces, and a few Configurable Functions have custom ID numbers (also called User ID numbers) that allow advanced users to identify specific datasets in different simulations where they could be used in different combinations.

During a simulation, each dataset for a Configurable Function, Path, or Road that has a custom ID will have only one instance of that dataset. If it reads a Parsfile that specifies an ID number that is already in use, it will override the data in memory. If it reads a Parsfile that specifies an ID number that is not yet in use, it will add the dataset and increase the number of datasets in use.

The advantage of using a custom ID is that it will be valid in different simulations. The path with user ID 1013 might be the third path defined in one simulation. In another simulation, it might be the seventh path. In both cases, the ID 1013 identify the same path.

Query ID Numbers

Query ID number are sometimes used by advanced users in some VS Command functions or VS API functions. The VS Command functions are documented in the last section of this document (page 71).

Coordinate Transformations

For a path to be used in a multibody physics math model, it is necessary to be able to perform two types of coordinate transformations:

1. calculate X and Y for a point given S and L, and
2. calculate S and L for a point given X and Y.

Transforming S-L to X-Y

Transforming from S-L to X-Y is accomplished with a direct sequence of calculations (no iteration required) for any VS Reference Path. Transformations of this sort are primarily used to locate and orient moving objects, and sometimes used to locate and orient the vehicle for initial conditions.

Transforming X-Y to S-L

Transforming from X-Y to S-L is needed to determine road properties where the tire/ground contacts occur, and to determine the relationship between points of interest and a target path for steering and speed controllers in the vehicle model.

Going from X-Y coordinates to S-L requires an iterative approach because the segment containing the X-Y pair must first be identified. VS Solvers keep track of the segment on which an object is located and use internal information to quickly locate the correct segment. As noted in the previous subsection, each query into a path has an associated query ID, such that when the query is repeated at the next time step, the VS Solver will start searching at the previous location associated with the internal query ID.

A significant complication arises when there are multiple pairs of S-L coordinates that would be valid for a single X-Y point of interest. For example, Figure 2 shows an aerial view of a racetrack Reference Path (blue) and a red point of interest. Any of the points labeled A through I on the Reference Path are related to the point of interest with a line perpendicular to the Reference Path. They therefore yield different, but valid, S-L coordinates for that X-Y point. In this case, point A is probably the desired X-Y location on the path with valid S-L coordinates.

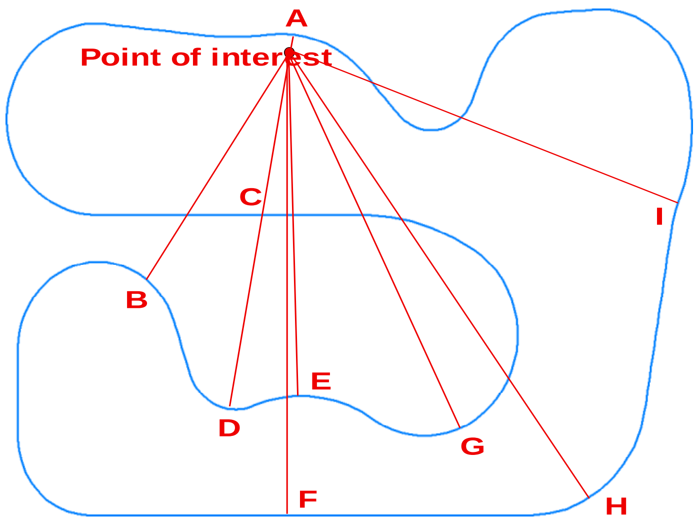


Figure 2. A Reference Path might have multiple valid S-L coordinates for a point of interest.

The internal query IDs usually resolve the ambiguity in S-L coordinate calculations. For example, suppose that the point of interest shown in Figure 2 is for a point of contact for the left-front tire of the simulated vehicle. When the S-L coordinates were calculated the previous time step, the point was very close to point A, and that station value would be the starting S coordinate used for computing the X-Y to S-L coordinate transformation. This resolves the ambiguity (in most reasonable cases) and reduces the number of iterations required to compute the S-L transformation.

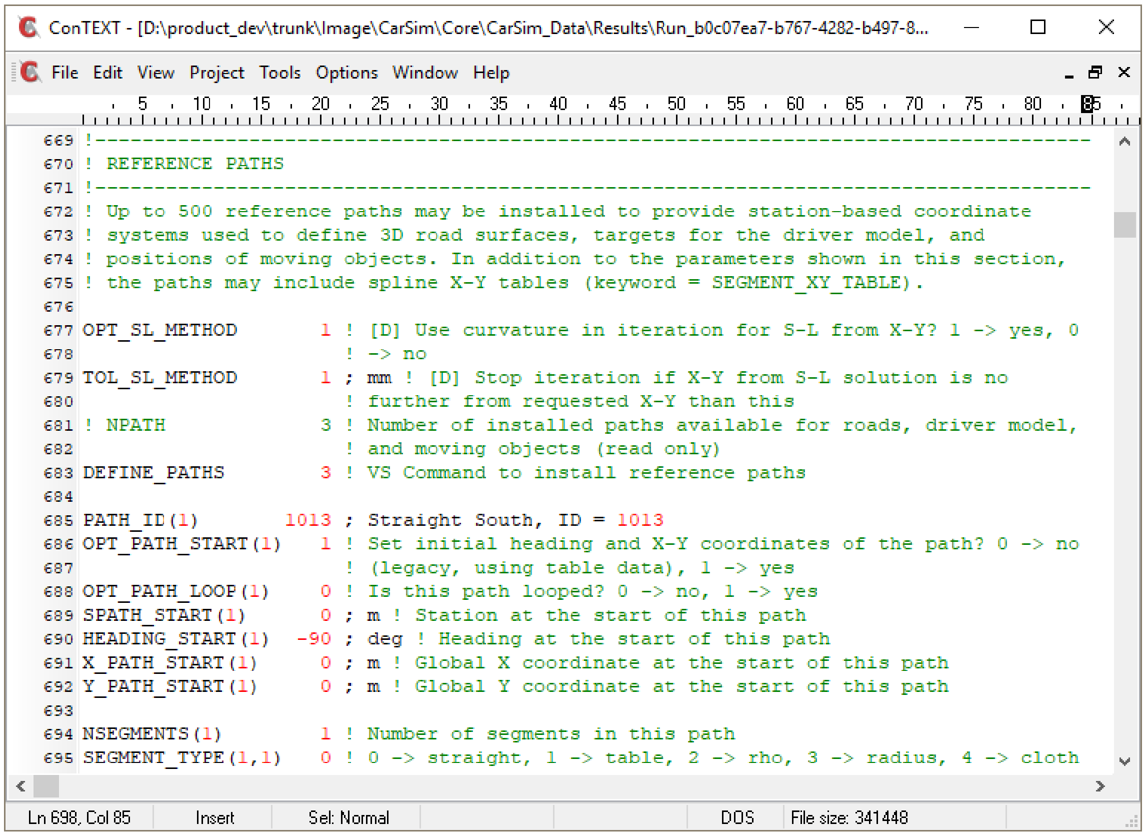
Regarding the iterative method itself, there are two approaches, based on the setting of the math model keyword `OPT_SL_METHOD` (zero or one). Both approaches use query IDs and operate in general as described above. The distinction is whether the path curvature is used in the iteration. When `OPT_SL_METHOD=1`, the estimate for station at any given iteration includes a dependence on path curvature. This can lead to more rapid convergence to a solution for station if the curvature is well-behaved. If the VS Math Model detects a failure to converge, the solution is re-attempted, this time not using path curvature. This is done automatically. When `OPT_SL_METHOD=0`, curvature is never used (the attempt using curvature is bypassed entirely).

The accuracy of the S-L from X-Y iteration may be fine tuned with the math model keyword `TOL_SL_METHOD`. This, in millimeters, sets the maximum distance between the requested X-Y and the X-Y implied by the S-L solution. In other words, the iteration halts when the distance is less than or equal to `TOL_SL_METHOD`, not when the distance is equal to this value.

Using VS Reference Paths

VS Reference Paths are used with closed-loop driver models and for controlling moving objects, such as traffic and ADAS sensor targets. They are also essential to the descriptions of VS Roads, as described in the next section.

Echo files are designed to function as inputs to VS Solvers, especially in the case where the Echo file written at the end of simulation is used later to continue the simulation. This requires that Reference Paths be shown in the Echo file before any references are made to their ID numbers for Road Surfaces, driver controllers, or any moving objects. Figure 3 shows the top of the Reference Paths section of an Echo file involving a simulation with three Reference Paths.


ConTEXT - [D:\product_dev\trunk\Image\CarSim\Core\CarSim_Data\Results\Run_b0c07ea7-b767-4282-b497-8...
File Edit View Project Tools Options Window Help
Ln 698, Col 85
Insert
Sel: Normal
DOS
File size: 341448

```
669 !------------------------------------------------------------------------------
670 ! REFERENCE PATHS
671 !------------------------------------------------------------------------------
672 ! Up to 500 reference paths may be installed to provide station-based coordinate
673 ! systems used to define 3D road surfaces, targets for the driver model, and
674 ! positions of moving objects. In addition to the parameters shown in this section,
675 ! the paths may include spline X-Y tables (keyword = SEGMENT_XY_TABLE).
676
677 OPT_SL_METHOD          1 ! [D] Use curvature in iteration for S-L from X-Y? 1 -> yes, 0
678                          ! -> no
679 TOL_SL_METHOD          1 ; mm ! [D] Stop iteration if X-Y from S-L solution is no
680                          ! further from requested X-Y than this
681 ! NPATH                3 ! Number of installed paths available for roads, driver model,
682                          ! and moving objects (read only)
683 DEFINE_PATHS           3 ! VS Command to install reference paths
684
685 PATH_ID(1)          1013 ; Straight South, ID = 1013
686 OPT_PATH_START(1)      1 ! Set initial heading and X-Y coordinates of the path? 0 -> no
687                          ! (legacy, using table data), 1 -> yes
688 OPT_PATH_LOOP(1)       0 ! Is this path looped? 0 -> no, 1 -> yes
689 SPATH_START(1)         0 ; m ! Station at the start of this path
690 HEADING_START(1)     -90 ; deg ! Heading at the start of this path
691 X_PATH_START(1)        0 ; m ! Global X coordinate at the start of this path
692 Y_PATH_START(1)        0 ; m ! Global Y coordinate at the start of this path
693
694 NSEGMENTS(1)           1 ! Number of segments in this path
695 SEGMENT_TYPE(1,1)      0 ! 0 -> straight, 1 -> table, 2 -> rho, 3 -> radius, 4 -> cloth
```

Figure 3. Echo file showing the Reference Paths section (straight path).

Note Every VS simulation includes at least one Reference Path. If the inputs to the math model do not include any paths, then a default path is automatically added during initialization that is a straight line oriented with a heading of zero, such that S = global X and L = global Y.

Reference Paths: IDs and Multiple Instances

In support of simulation scenarios involving multiple paths, each path includes a custom ID specified with the parameter `PATH_ID` (line 685). The path used for a road, driver model, or moving object is specified using this ID number (e.g., 1013 in this example).

When setting up a Reference Path in the Browser GUI, there are usually two options for setting the ID (Figure 4). If the option is selected to set the ID automatically, it simply matches the internal index number, e.g., `PATH_ID(2)` would be set to 2. If the option is selected to set a custom ID, a yellow field is displayed for an integer value of 999 or greater (Figure 4 and line 685 in Figure 3 both show an ID set to 1013).

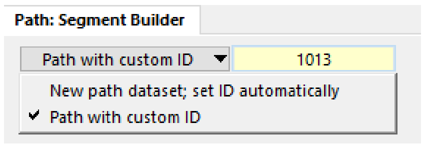


Figure 4. When describing a Reference Path in the GUI, there are two options to set the ID.

When linking to a dataset from one of the Reference Path libraries in the Browser, a new path is added to the model unless two conditions are satisfied:

1. The path has a custom `PATH_ID` (999 or larger), and
2. A Reference Path already exists with that ID.

If a Reference Path with the custom ID already exists, the information from the linked dataset is used to replace the path description.

Paths, Roads, and a few Configurable Functions that include custom IDs typically use a VS Command (written to the Parsfile for the dataset) to setup the Configurable Function dataset.

For paths, the VS Command `SET_IPATH_FOR_ID` checks to see if a requested ID is already in use. If it is, the data from the screen will reset the properties of that existing dataset. If not, a new dataset is added to the model. (This command is described along with other `SET_` commands later in this document, on page 71.)

The requested ID can be specified with an integer or with the name of a parameter created elsewhere with the `DEFINE_PARAMETER` VS Command. If the ID is a number (just digits 0-9), it is checked by the GUI to ensure the value is either 0 or greater than or equal to 999.

If the ID is a symbol or symbolic formula, it will not be checked by the GUI. However, it is checked when the dataset is used in a simulation, and an error will be generated if the value is not valid.

The number of paths used in a simulation is set to an internal parameter `NPATH`, shown in the Echo File (line 681, Figure 3).

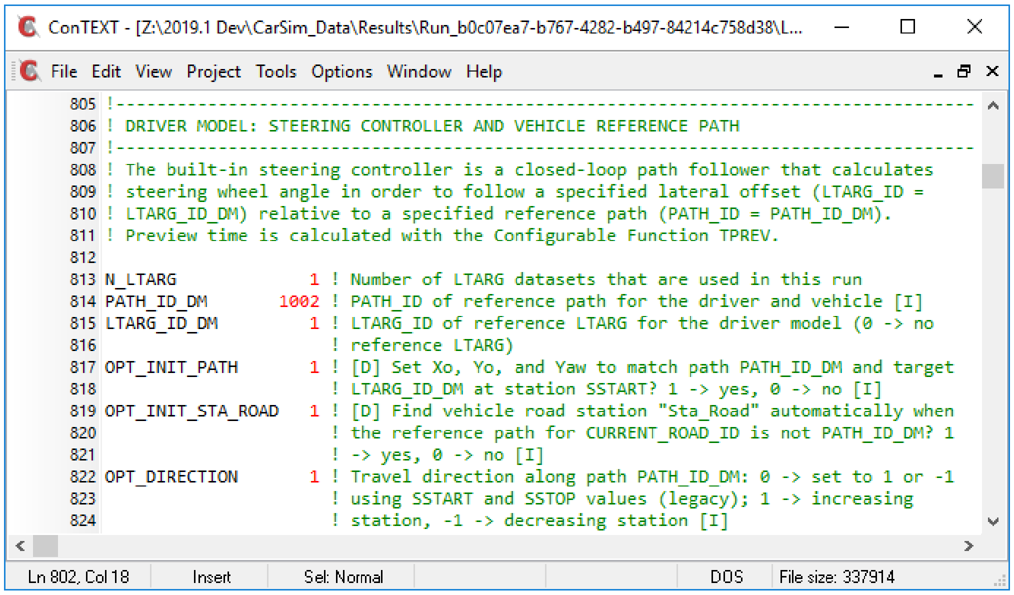
Many vehicle tests involving a single road surface and single path, and in these cases the automatic ID works well. However, ADAS scenarios often involve multiple paths and/or roads. In these cases, custom IDs allow datasets to be reused in different scenarios. For example, if `PATH_ID(3) = 1234`, then the ID number `1234` is used in simulations regardless of when the path was defined relative to other paths. (It might not be the third path when linked to a different simulation.) In another simulation, the same Reference Path might be the ninth loaded (`PATH_ID(9) = 1234`). The custom ID is unchanged, so references made to that ID number for the driver or moving objects can still be valid.

Reference Paths: Driver and Rider Models

One of the Reference Paths is identified as the driver/rider Reference Path. It is the path chosen for the internal driver/rider model if the controller is active and is identified in the VS Solver with the `PATH_ID_DM` parameter (line 814, Figure 5).

Even when the internal path-follower driver model is not active, the selected path is used to define path-related output variables for the vehicle such as `Station`. The path with ID = `PATH_ID_DM` may also be used to set the initial position and yaw angle for the vehicle, depending on the value of the parameter `OPT_INIT_PATH` (line 817). (The initialization options are described in a later section, starting on page 63.)

Every GUI library screen used to define a path writes a Parsfile that includes the statement "`PATH_ID_DM = PATH_ID`". As a result, `PATH_ID_DM` is always set to use the dataset that was read most recently by the VS Solver. To choose another path for `PATH_ID_DM`, assign a different value to the `PATH_ID_DM` parameter in a Miscellaneous yellow field that is read by the VS Solver after all the paths have been defined.


ConTEXT - [Z:\2019.1 Dev\CarSim_Data\Results\Run_b0c07ea7-b767-4282-b497-84214c758d38\L...

```
805 !------------------------------------------------------------------------------
806 ! DRIVER MODEL: STEERING CONTROLLER AND VEHICLE REFERENCE PATH
807 !------------------------------------------------------------------------------
808 ! The built-in steering controller is a closed-loop path follower that calculates
809 ! steering wheel angle in order to follow a specified lateral offset (LTARG_ID =
810 ! LTARG_ID_DM) relative to a specified reference path (PATH_ID = PATH_ID_DM).
811 ! Preview time is calculated with the Configurable Function TPREV.
812
813 N_LTARG                1 ! Number of LTARG datasets that are used in this run
814 PATH_ID_DM          1002 ! PATH_ID of reference path for the driver and vehicle [I]
815 LTARG_ID_DM            1 ! LTARG_ID of reference LTARG for the driver model (0 -> no
816                          ! reference LTARG)
817 OPT_INIT_PATH          1 ! [D] Set Xo, Yo, and Yaw to match path PATH_ID_DM and target
818                          ! LTARG_ID_DM at station SSTART? 1 -> yes, 0 -> no [I]
819 OPT_INIT_STA_ROAD      1 ! [D] Find vehicle road station "Sta_Road" automatically when
820                          ! the reference path for CURRENT_ROAD_ID is not PATH_ID_DM? 1
821                          ! -> yes, 0 -> no [I]
822 OPT_DIRECTION          1 ! Travel direction along path PATH_ID_DM: 0 -> set to 1 or -1
823                          ! using SSTART and SSTOP values (legacy); 1 -> increasing
824                          ! station, -1 -> decreasing station [I]
```

Figure 5. Top of the driver model section of the Echo file with Path information.

Moving Objects

VS Math Models include up to 200 objects whose locations and motions are independent of the simulated ego vehicle. Each object has a set of properties, Output variables, and Import variables as described in the **Help** document *ADAS Sensors and Moving Objects*.

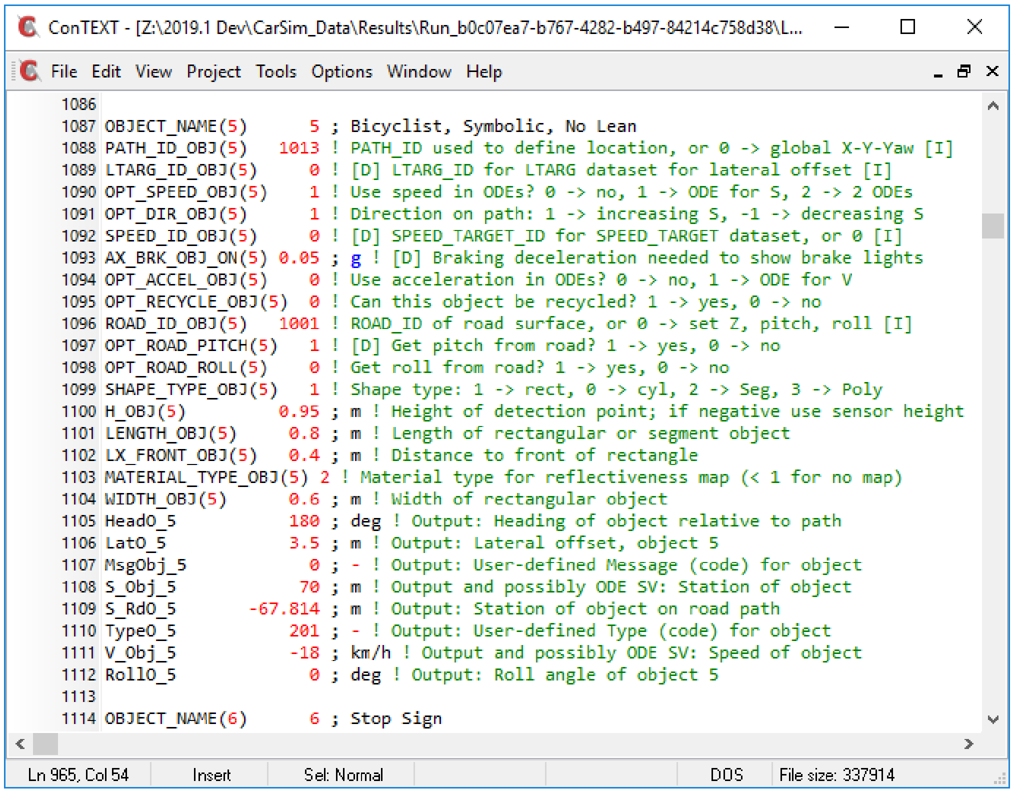
Three of the parameters for each object involve setting its location and orientation using paths and roads. These are shown in the Moving Objects section of the Echo file (Figure 6).

1. `PATH_ID_OBJ`: The object can be located horizontally using a Reference Path, where the path is identified with the parameter `PATH_ID_OBJ` (line 1088). If `PATH_ID_OBJ` = 0 then the object is located using global X and Y coordinates that are.

2. `LTARG_ID_OBJ`: The object can include a lateral offset calculated automatically with the `LTARG` Configurable Function, using a dataset ID `LTARG_ID_OBJ`. Specify an `LTARG_ID_OBJ` = 0 to disable this option (line 1089). (The `LTARG` function is described in a later subsection, page 25). This option is only available when `PATH_ID_OBJ` > 0.

3. `ROAD_ID_OBJ`: The object's elevation and 3D orientation (pitch and roll) can be calculated automatically using a 3D road surface. A road is specified using the parameter `ROAD_ID_OBJ` (line 1096); this is the custom ID for the road `ROAD_ID`, as described in the next section (page 26).

4. If the horizontal position of the moving object is based on a Reference Path (`PATH_ID_OBJ` > 0), then the initial station for the object on that path is written in the Echo file (line 1108, this is output variable `S_Obj_5` for object #5). If the vertical position of the moving object is based on a road (`ROAD_ID_OBJ` > 0) and the Reference Path for the road (described later) is not `PATH_ID_OBJ`, then the VS Solver keeps track of the station of the object on the road Reference Path. That station, `S_RdO_5` for object #5, is also written in the Echo file (line 1109).


ConTEXT - [Z:\2019.1 Dev\CarSim_Data\Results\Run_b0c07ea7-b767-4282-b497-84214c758d38\L...
File Edit View Project Tools Options Window Help
Ln 965, Col 54
Insert
Sel: Normal
DOS
File size: 337914

```
1086
1087 OBJECT_NAME(5)      5 ; Bicyclist, Symbolic, No Lean
1088 PATH_ID_OBJ(5)   1013 ! PATH_ID used to define location, or 0 -> global X-Y-Yaw [I]
1089 LTARG_ID_OBJ(5)     0 ! [D] LTARG_ID for LTARG dataset for lateral offset [I]
1090 OPT_SPEED_OBJ(5)    1 ! Use speed in ODEs? 0 -> no, 1 -> ODE for S, 2 -> 2 ODEs
1091 OPT_DIR_OBJ(5)      1 ! Direction on path: 1 -> increasing S, -1 -> decreasing S
1092 SPEED_ID_OBJ(5)     0 ! [D] SPEED_TARGET_ID for SPEED_TARGET dataset, or 0 [I]
1093 AX_BRK_OBJ_ON(5) 0.05 ; g ! [D] Braking deceleration needed to show brake lights
1094 OPT_ACCEL_OBJ(5)    0 ! Use acceleration in ODEs? 0 -> no, 1 -> ODE for V
1095 OPT_RECYCLE_OBJ(5)  0 ! Can this object be recycled? 1 -> yes, 0 -> no
1096 ROAD_ID_OBJ(5)   1001 ! ROAD_ID of road surface, or 0 -> set Z, pitch, roll [I]
1097 OPT_ROAD_PITCH(5)   1 ! [D] Get pitch from road? 1 -> yes, 0 -> no
1098 OPT_ROAD_ROLL(5)    0 ! Get roll from road? 1 -> yes, 0 -> no
1099 SHAPE_TYPE_OBJ(5)   1 ! Shape type: 1 -> rect, 0 -> cyl, 2 -> Seg, 3 -> Poly
1100 H_OBJ(5)         0.95 ; m ! Height of detection point; if negative use sensor height
1101 LENGTH_OBJ(5)     0.8 ; m ! Length of rectangular or segment object
1102 LX_FRONT_OBJ(5)   0.4 ; m ! Distance to front of rectangle
1103 MATERIAL_TYPE_OBJ(5) 2 ! Material type for reflectiveness map (< 1 for no map)
1104 WIDTH_OBJ(5)      0.6 ; m ! Width of rectangular object
1105 HeadO_5           180 ; deg ! Output: Heading of object relative to path
1106 LatO_5            3.5 ; m ! Output: Lateral offset, object 5
1107 MsgObj_5            0 ; - ! Output: User-defined Message (code) for object
1108 S_Obj_5            70 ; m ! Output and possibly ODE SV: Station of object
1109 S_RdO_5       -67.814 ; m ! Output: Station of object on road path
1110 TypeO_5           201 ; - ! Output: User-defined Type (code) for object
1111 V_Obj_5           -18 ; km/h ! Output and possibly ODE SV: Speed of object
1112 RollO_5             0 ; deg ! Output: Roll angle of object 5
1113
1114 OBJECT_NAME(6)      6 ; Stop Sign
```

Figure 6. Description of a moving object using path and road information.

In some cases, there is a requirement to control the 3D motion and orientation of the moving objects entirely with VS Commands or Import variables from external models in Simulink, etc. See the **Help** document *ADAS Sensors and Moving Objects* for the full list of import variables available for moving objects.

Defining Reference Paths

The VS Browser includes eight libraries for defining paths. The modeling options in VehicleSim products have been extended from a single Path (1996), to two Paths (one for the driver and one for the road, added in 2011), to the current version, where up to 500 Reference Paths can be defined (starting in 2019.1). All Path and Road libraries are accessed from the submenu **Libraries** > **Paths and Road Surfaces**.

Path: Segment Builder. Use this screen to specify a path using a sequence of segments, where each segment can be a straight line, a circular arc, a table of X-Y coordinates, or a clothoid (Euler spiral) transition.

Path: X-Y Coordinates. Use this screen to define a path using a single segment represented by a table of X-Y coordinates.

Path: X-Y Coordinates for Segment. Use this screen to define a table of X-Y coordinates for use as a segment in a path assembled with the **Path: Segment Builder** screen. This is the only Path screen that cannot be used on its own (i.e., linked directly to a **Road: 3D Surface** screen).

Path: X-Y Coordinates (Legacy). This screen is identical in function to the **Path: X-Y Coordinates** screen; it exists to support legacy datasets.

Path/Road: Segment Builder (Legacy). Use this screen to specify a path and associated road surface using a sequence of straight lines and arcs. A drop-down control determines whether the dataset is for a path by itself, or if it also includes some road elevation and/or friction properties. Option also exist to include some road properties (elevation, banking, and friction).

Road: **X-Y-Z Coordinates of Reference Line**. Use this screen to define a path using a single segment represented by a table of X-Y-Z coordinates.

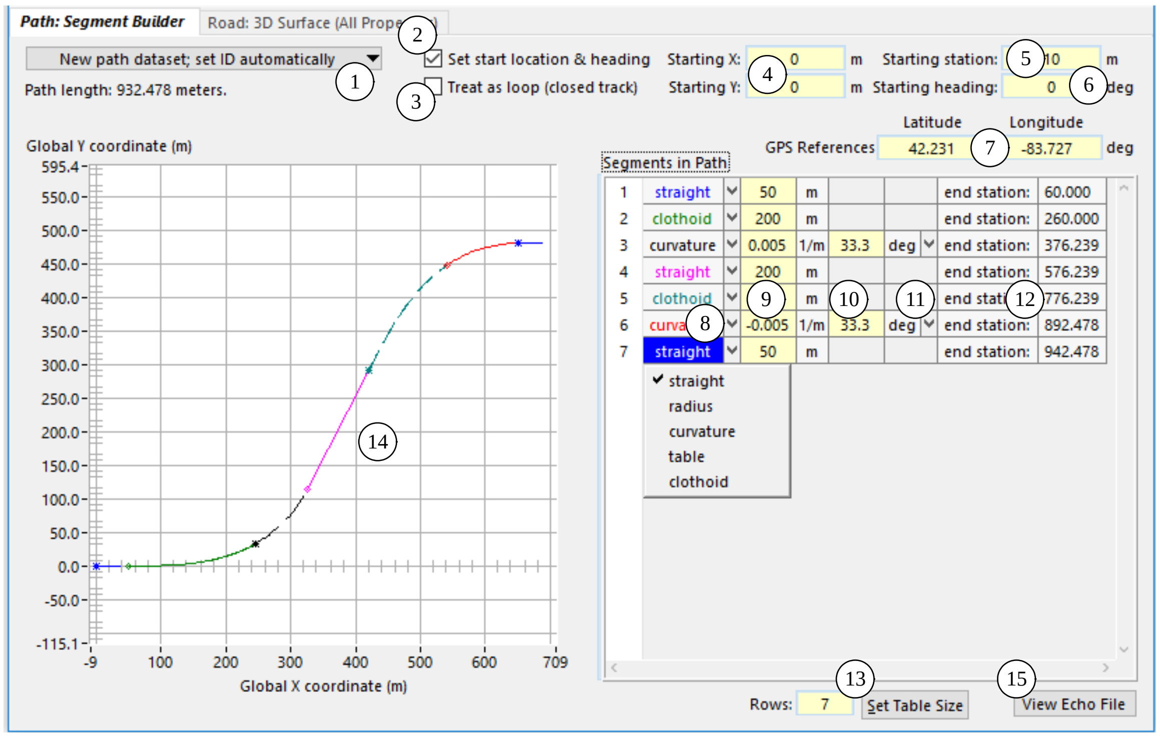
Road: **X-Y-Z Coordinates of Edges**. Use this screen to define a Reference Path, vertical geometry, and cross-slope (banking) of a road from global X-Y-Z coordinates for two paths (an inner path and an outer path).

Scene: External Import. Use this screen to import GPS data from atlas.carsim.com or ADAS RP, or VS Paths integrated with animator assets from VS Scene Builder. This screen and its functions are described in a document: *Scene External Import* (**Help** > **Paths, Road Surfaces, and Scenes**).

Of these eight libraries, the ones most used to define a path are **Path: Segment Builder**, **Path: X-Y Coordinates**, and **Scene: External Import**.

The Path: Segment Builder Screen

The **Path: Segment Builder** screen (Figure 7) provides the most comprehensive tool for assembling a Reference Path.



New path dataset; set ID automatically

Path length: 932.478 meters.

☑ Set start location & heading

☐ Treat as loop (closed track)

Starting X: 0 m

Starting Y: 0 m

Starting station: 10 m

Starting heading: 0 deg

	Latitude	Longitude	
GPS References	42.231	-83.727	deg

1	straight	50	m			end station:	60.000
2	clothoid	200	m			end station:	260.000
3	curvature	0.005	1/m	33.3	deg	end station:	376.239
4	straight	200	m			end station:	576.239
5	clothoid		m			end stati	776.239
6	curva	-0.005	1/m	33.3	deg	end station:	892.478
7	straight	50	m			end station:	942.478

Rows: 7 Set Table Size View Echo File

Figure 7. The Path: Segment Builder screen.

The path is built from segments starting with an initial global location (X and Y coordinates (4)) and heading angle (6). Each segment is defined with a type and one or two descriptive parameters. In assembling the path, the VS Solver maintains continuity such that the starting location and heading of each new segment match the ending location and heading of the previous segment.

Path Parameters

The controls at the top of the screen are used to set parameters for the entire path.

(1) Drop-down list to specify how this dataset will be identified. The control offers the two options that were described earlier (Figure 4, page 7). The ID can be set automatically (Figure 7), or a custom ID can be set (e.g., 1013 in Figure 4). If a custom ID is specified, the value is checked as described earlier (page 7).

(2) Checkbox to show fields for setting the initial coordinates (4) and heading angle (6). If not checked, the path starts at (0,0), pointed east (heading = 0).

(3) Checkbox for the path loop option (keyword = `OPT_PATH_LOOP`). This determines how a VS Solver handles X-Y and S-L transforms outside the range of station covered by the table. When not checked, the path extends infinitely in both directions, with zero curvature outside the range of station covered in the table. When checked, the first and last X-Y coordinates at the end of the path must be the same as the starting X-Y coordinates (within a tolerance), and the heading must match (within a tolerance). The VS Solver will generate an error if the initial and final X, Y, and Heading values are not within a tolerance built into the solvers. If this happens, you can view the Echo file generated for the path (15) to see details of the X, Y, and Heading values after each segment.

(4) X and Y global coordinates for the path at the start of the first segment (keywords = `X_PATH_START`, `Y_PATH_START`). These are not shown unless the box (2) is checked.

(5) Starting station (keyword = `SPATH_START`). This defines the station at the beginning of the Reference Path.

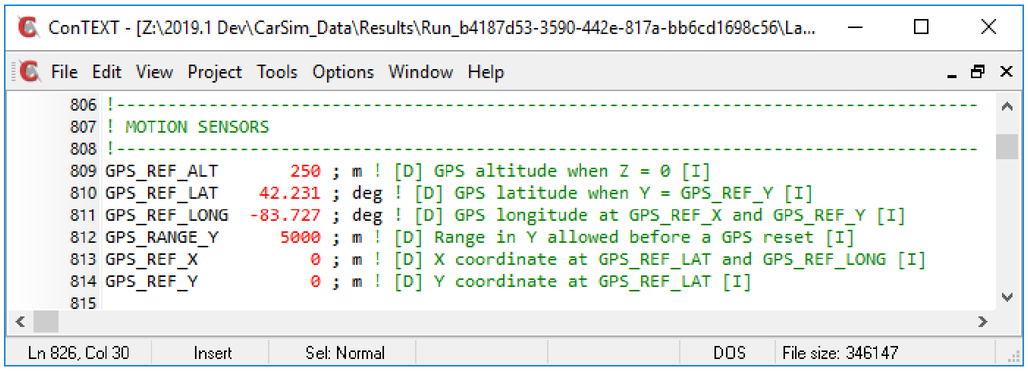
(6) Heading angle at the start of the path (keyword = `HEADING_START`). This is not shown unless the box (2) is checked.

Support of GPS

CarSim, TruckSim, and BikeSim support GPS output variables for tracking motion; additionally, some of the other Reference Path screens allow import of GPS data to obtain X-Y coordinates for a path. Some of the parameters associated with GPS output are supported on this screen. Figure 8 shows the top of the Motion Sensor section of the Echo file, listing all the parameters that define how the global X-Y-Z multibody coordinates system is linked to GPS. Of interest here are those for reference latitude and longitude.

(7) GPS Reference Coordinates: Latitude and Longitude, specified in degrees (keywords = `GPS_REF_LAT` and `GPS_REF_LONG`). These are the GPS coordinates at a point in the global coordinate system specified by the parameters `GPS_REF_X` and `GPS_REF_Y`. Unless specified otherwise, the reference point coordinates retain their default values `GPS_REF_X` = 0 and `GPS_REF_Y` = 0.

The VS Solver uses these four parameters to calculate GPS latitude and longitude output variables `GPS_LatA` and `GPSLongA` from global X (east) and global Y (north) coordinates.


ConTEXT - [Z:\2019.1 Dev\CarSim_Data\Results\Run_b4187d53-3590-442e-817a-bb6cd1698c56\La...
File Edit View Project Tools Options Window Help

```
!------------------------------------------------------------------------------
! MOTION SENSORS
!------------------------------------------------------------------------------
GPS_REF_ALT       250 ; m ! [D] GPS altitude when Z = 0 [I]
GPS_REF_LAT    42.231 ; deg ! [D] GPS latitude when Y = GPS_REF_Y [I]
GPS_REF_LONG  -83.727 ; deg ! [D] GPS longitude at GPS_REF_X and GPS_REF_Y [I]
GPS_RANGE_Y      5000 ; m ! [D] Range in Y allowed before a GPS reset [I]
GPS_REF_X           0 ; m ! [D] X coordinate at GPS_REF_LAT and GPS_REF_LONG [I]
GPS_REF_Y           0 ; m ! [D] Y coordinate at GPS_REF_LAT [I]
```

Ln 826, Col 30
Insert
Sel: Normal
DOS
File size: 346147

Figure 8. Echo file for Motion Sensors, including GPS.

When there are multiple paths used in a simulation and each have a specification for the GPS Reference Coordinates, the GPS Reference Coordinates that are specified on the last path processed by the VS Solver are the set used for the simulation.

For more information about how GPS coordinates are handled in VS Math Models, please see the Tech Memo *GPS Coordinates* from the **Help** menu. The use of GPS data to create Reference Paths or segments is described in the subsection **Importing GPS Data** (page 22).

Table of Segment Properties

A table with the title **Segments in Path** builds the path from segments, with one segment per row.

(8) The first item in each row is a drop-down control to select the type of segment:

1. Straight line.
2. Circular arc, defined with radius.
3. Circular arc, defined with curvature (inverse radius).
4. Table of X-Y coordinates that are fitted using spline interpolation. The table is provided with a linked dataset from the **Path: X-Y Coordinates for Segment** library.
5. Clothoid (Euler spiral) transition between two other segments, defined with three parameters.

(9) The first parameter after the type of segment depends on the type of segment.

1. If the option is chosen to use a table, then this field has a blue data link that goes to the **Path: X-Y Coordinates for Segment** library described later (page 20).
2. If the type is a straight line or a clothoid transition, then the parameter is a length, in meters.
3. If the type is a circular arc, then the parameter is either a radius in meters or curvature in 1/m.

If the turn is to the left, the radius or curvature value is positive; if the turn is to the right, the radius or curvature value is negative.

(10) If the segment type is a circular arc (radius or curvature), a second parameter is needed to specify a central angle or arc length with units specified using the next control (11).

(11) Two options are available for specifying an arc length: central angle (deg) or length (m).

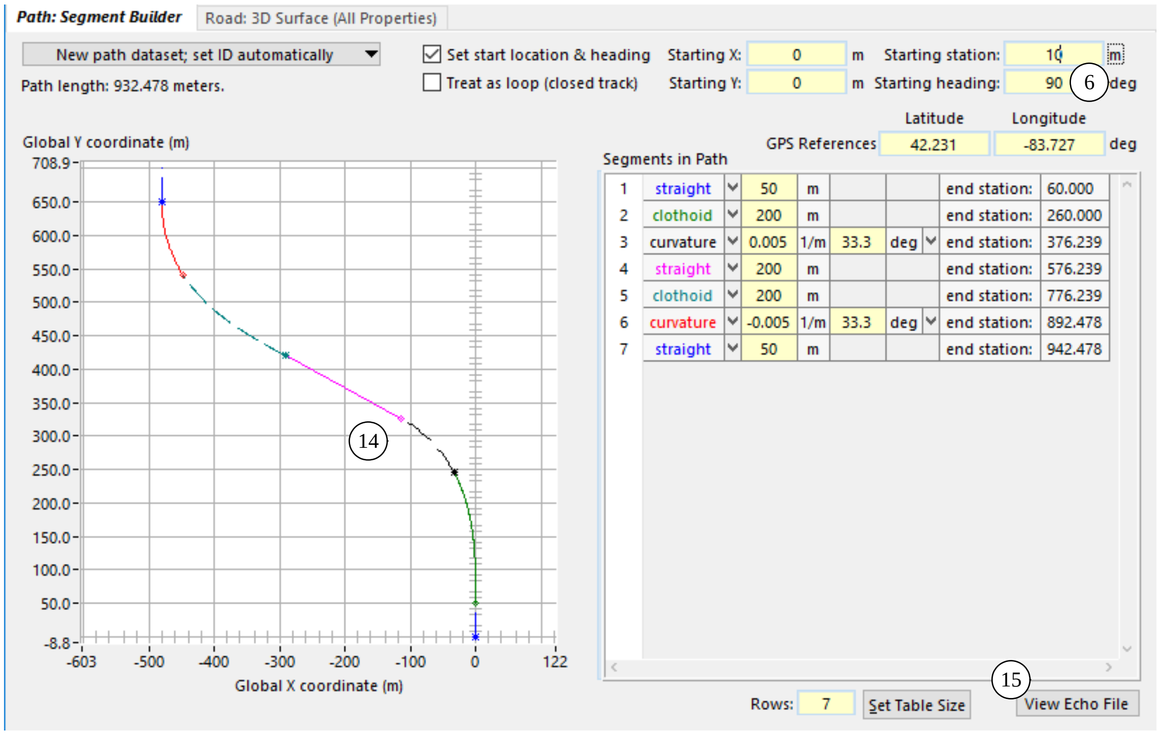
(12) The station at the end of each segment is calculated and shown in the table.

(13) Use the yellow field to specify the number of segments in the path, then click the adjacent button to resize the table. VS Solvers support up to 100 segments per path.

Visualization of the Path

(14) A plot is made with coordinates calculated using the current settings on the screen. Different segments are shown in different plot colors and/or line styles that match the colors used for the first text column of the table (8).

Whenever a change is made to the specifications of the path, the VS Browser generates new X-Y coordinates of points along the path and updates the plot. For example, Figure 9 shows how the path is cleanly rotated by changing the initial heading angle from 0 to 90° (6).



Path length: 932.478 meters.

[✓] Set start location & heading
[ ] Treat as loop (closed track)

Starting X: 0 m
Starting Y: 0 m
Starting station: 10 m
Starting heading: 90 deg

	Latitude	Longitude	
GPS References	42.231	-83.727	deg

Segments in Path

1	straight	50	m			end station:	60.000
2	clothoid	200	m			end station:	260.000
3	curvature	0.005	1/m	33.3	deg	end station:	376.239
4	straight	200	m			end station:	576.239
5	clothoid	200	m			end station:	776.239
6	curvature	-0.005	1/m	33.3	deg	end station:	892.478
7	straight	50	m			end station:	942.478

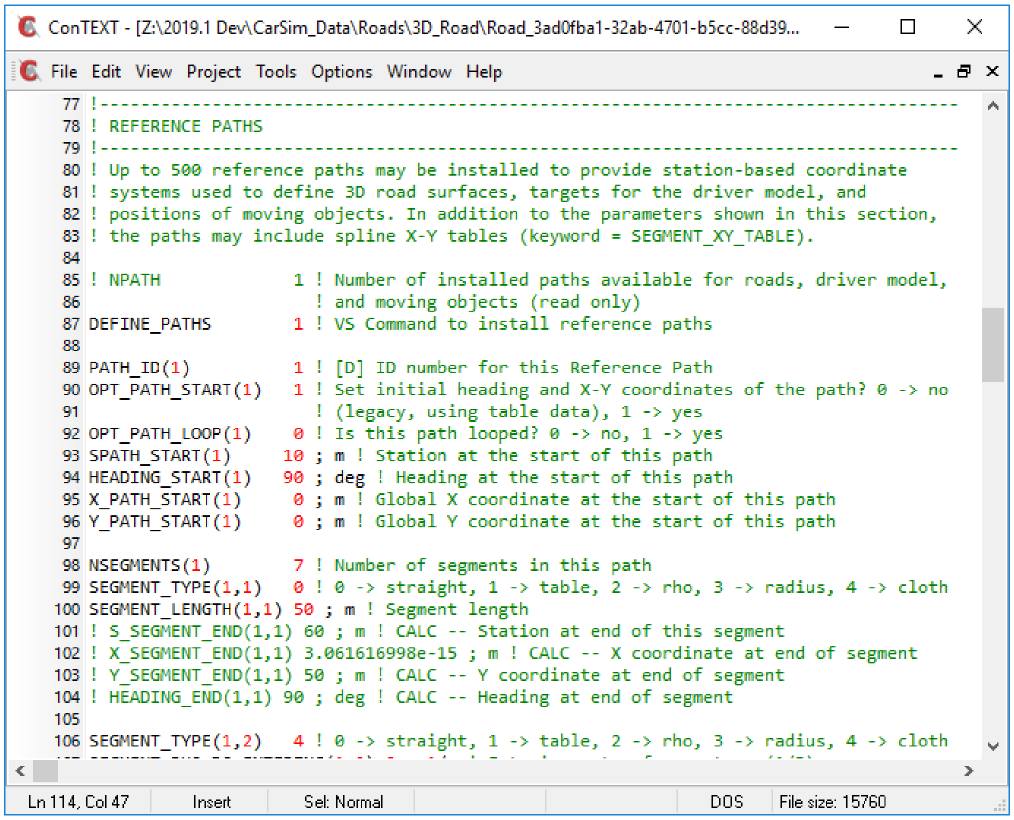
Rows: 7

Figure 9. Effect of setting the starting heading angle to 90° for the example path.

The Echo File and Road Calculator

This screen makes use of a special VS Solver called VS Road Calculator. Unlike most other VS Solvers, it does not contain a vehicle math model, and is not used to run time-domain simulations. Instead, it supports most road and path commands and data types that are available in the VS Solvers for the vehicle math models. The VS Browser uses the Road Calculator to perform path and road-based calculations via the VS API.

15 As with any VS Solver, an Echo file is generated whenever VS Road Calculator is used to update information on the screen. Use this button to view the file (Figure 10). The Echo file lists the following:



```
 77 !------------------------------------------------------------------------------------
 78 ! REFERENCE PATHS
 79 !------------------------------------------------------------------------------------
 80 ! Up to 500 reference paths may be installed to provide station-based coordinate
 81 ! systems used to define 3D road surfaces, targets for the driver model, and
 82 ! positions of moving objects. In addition to the parameters shown in this section,
 83 ! the paths may include spline X-Y tables (keyword = SEGMENT_XY_TABLE).
 84
 85 ! NPATH                1 ! Number of installed paths available for roads, driver model,
 86                          ! and moving objects (read only)
 87 DEFINE_PATHS           1 ! VS Command to install reference paths
 88
 89 PATH_ID(1)             1 ! [D] ID number for this Reference Path
 90 OPT_PATH_START(1)      1 ! Set initial heading and X-Y coordinates of the path? 0 -> no
 91                          ! (legacy, using table data), 1 -> yes
 92 OPT_PATH_LOOP(1)       0 ! Is this path looped? 0 -> no, 1 -> yes
 93 SPATH_START(1)        10 ; m ! Station at the start of this path
 94 HEADING_START(1)      90 ; deg ! Heading at the start of this path
 95 X_PATH_START(1)        0 ; m ! Global X coordinate at the start of this path
 96 Y_PATH_START(1)        0 ; m ! Global Y coordinate at the start of this path
 97
 98 NSEGMENTS(1)           7 ! Number of segments in this path
 99 SEGMENT_TYPE(1,1)      0 ! 0 -> straight, 1 -> table, 2 -> rho, 3 -> radius, 4 -> cloth
100 SEGMENT_LENGTH(1,1) 50 ; m ! Segment length
101 ! S_SEGMENT_END(1,1) 60 ; m ! CALC -- Station at end of this segment
102 ! X_SEGMENT_END(1,1) 3.061616998e-15 ; m ! CALC -- X coordinate at end of segment
103 ! Y_SEGMENT_END(1,1) 50 ; m ! CALC -- Y coordinate at end of segment
104 ! HEADING_END(1,1) 90 ; deg ! CALC -- Heading at end of segment
105
106 SEGMENT_TYPE(1,2)      4 ! 0 -> straight, 1 -> table, 2 -> rho, 3 -> radius, 4 -> cloth
```

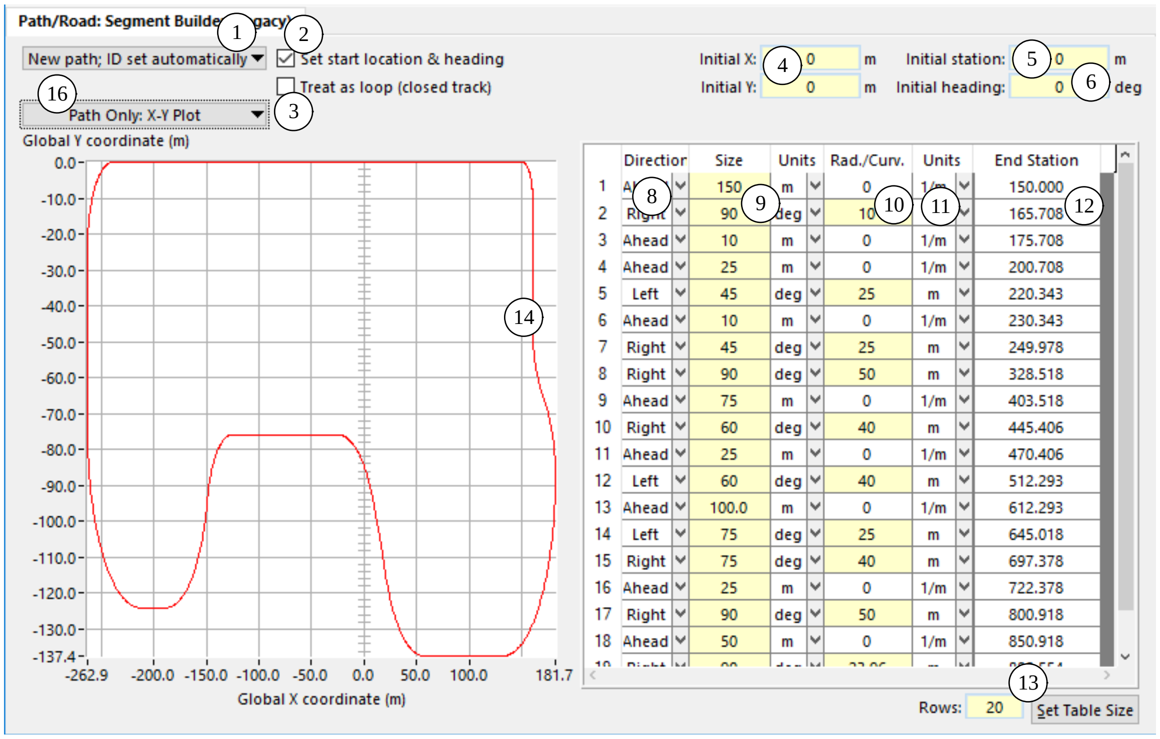
Figure 10. Echo file for the example path, generated by VS Road Calculator.

1. Type of each segment (lines 99, 106) along with the parameters associated with the type (e.g., radius, central angle, length).
2. Station at the end of each segment (line 101).
3. Global X and Y coordinates at the end of each segment (lines 102, 103).
4. Heading angle at the end of each segment (line 104).

The Echo file is rewritten whenever a change is made in the table in the GUI. If you leave the Echo file in view with the text editor, it will refresh whenever changes are made to show the results immediately (just click in the text window to trigger a refresh).

The Path/Road: Segment Builder (Legacy) Screen

The **Path/Road: Segment Builder (Legacy)** screen (Figure 11) was introduced in 2008 (CarSim 7.1). At that time, the screen served as a tool to build the path used for the single road in the simulation with segments, which were used to write a single table of X-Y coordinates for the VS Solver.



☑ Set start location & heading
☐ Treat as loop (closed track)

Initial X: 0 m Initial station: 0 m
Initial Y: 0 m Initial heading: 0 deg

	Directior	Size	Units	Rad./Curv.	Units	End Station
1	Ahead	150	m	0	1/m	150.000
2	Right	90	deg	10	m	165.708
3	Ahead	10	m	0	1/m	175.708
4	Ahead	25	m	0	1/m	200.708
5	Left	45	deg	25	m	220.343
6	Ahead	10	m	0	1/m	230.343
7	Right	45	deg	25	m	249.978
8	Right	90	deg	50	m	328.518
9	Ahead	75	m	0	1/m	403.518
10	Right	60	deg	40	m	445.406
11	Ahead	25	m	0	1/m	470.406
12	Left	60	deg	40	m	512.293
13	Ahead	100.0	m	0	1/m	612.293
14	Left	75	deg	25	m	645.018
15	Right	75	deg	40	m	697.378
16	Ahead	25	m	0	1/m	722.378
17	Right	90	deg	50	m	800.918
18	Ahead	50	m	0	1/m	850.918

Rows: 20

Figure 11. The Path / Road: Segment Builder (Legacy) screen.

This screen was modified in versions 9 (2015) and 2016 to provide more options and better accuracy. Rather than generating a Parsfile with X-Y coordinates, the screen now copies the segment definitions (straight lines and circular arcs) using keywords that are recognized by the VS Solvers. This is the same method used to generate the Parsfile for the **Path: Segment Builder** screen, described in the previous subsection.

One change made between version 9 and 2016 involves the use of the custom Path ID, which is now handled the same as in the **Path: Segment Builder** screen: if a path exists with the specified ID, it is updated with the data from the screen; if no existing path has the specified ID, a new one is created.

Most of the controls related to setting the path were described in the previous subsection; content that is unique to the **Path/Road: Segment Builder (Legacy)** is described below. Controls involving road elevation and friction properties are described later, in the *Road Surface Physical Properties* section (page 44).

Controls shared with the Path: Segment Builder Screen

The circled numbers in Figure 11 going from ① to ⑭ identify controls that also exist on the **Path: Segment Builder** screen (Figure 7, page 11), described previously.

There are a few differences in two of these controls:

⑧ Column of segment types. This is similar to the control for specifying the segment type on the **Path: Segment Builder** screen. However, in the case of the **Path/Road: Segment Builder (Legacy)** screen, only three options are available: **Left**, **Ahead**, and **Right**. If either **Left** or **Right** is selected, then the segment is a circular arc. If **Ahead** is selected, then the segment is a straight line extending continuously from the end of the previous segment. Clothoids and X-Y spline tables are not supported in this legacy library screen.

(9) Radius or curvature. If the segment is curved, then this column specifies the radius or the curvature (inverse radius) of the segment. The choice of radius (m) or curvature (1/m) is set with the adjacent drop-down control. If the segment is a straight line, then the curvature is ignored.

Control unique to this screen

The control (16) is unique for this screen.

(16) Drop-down list to specify the information shown on the screen, and whether the dataset includes any road properties (Figure 12).

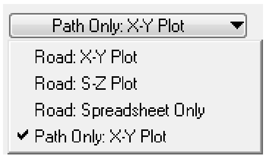


Figure 12. Options for showing plots and limiting the scope of the dataset.

The first three options offer support for a path used in a 3D road, with friction and elevation information (described later in the Road Surface section, page 44). The fourth option limits the scope of the dataset to a Reference Path. When this option is selected, only controls related to a Reference Path are shown (Figure 11).

Path and Road Tables with X-Y Coordinates

Four screens are available to define tables of global X and Y coordinates.

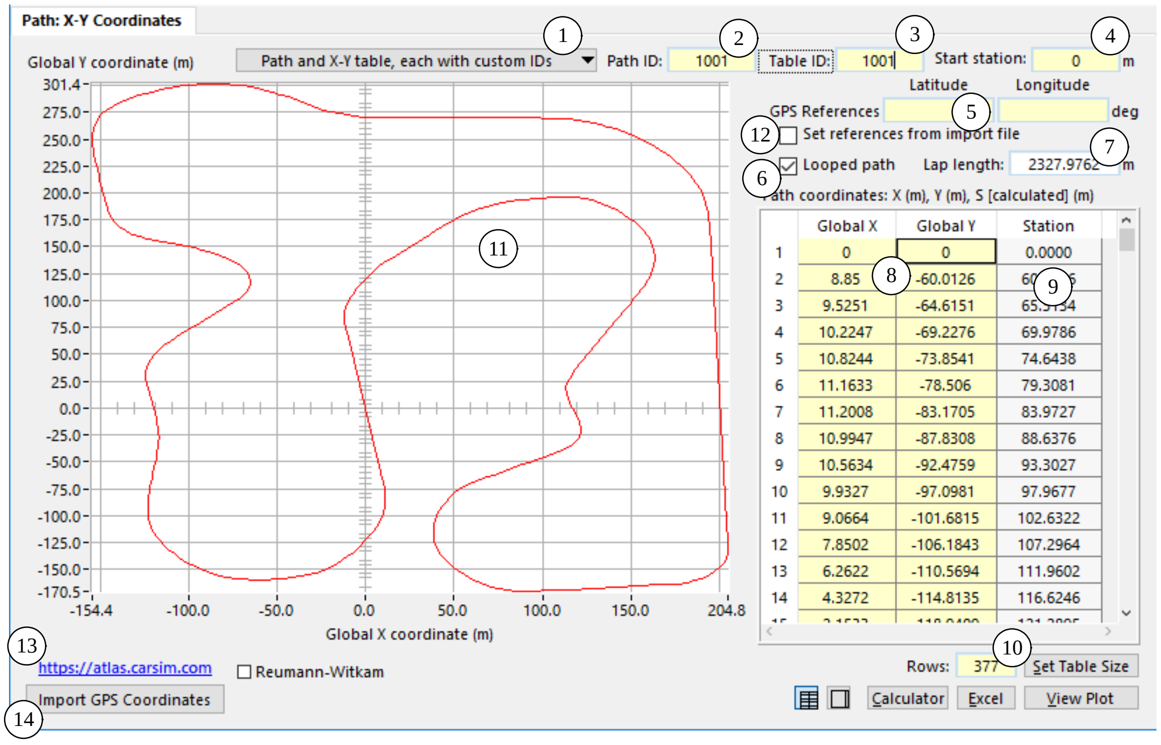
1. **Path: X-Y Coordinates**
2. **Path: X-Y Coordinates for Segment**
3. **Road: X-Y-Z Coordinates of Reference Line**
4. **Road: X-Y-Z Coordinates of Edges**

The first two are described in this section. The two screens that start with **Road** include road elevation properties and are described later in the Road Surfaces section (page 48).

Path: X-Y Coordinates

Figure 13 shows the **Path: X-Y Coordinates** screen. Datasets from this library can be used to define a Reference Path using a table of X-Y coordinates, where the path is either the Reference Path for a road, or is used as an additional path to an existing road for use by the Closed-Loop Path-Following steering controller, moving objects, etc.

The numbered controls are used to specify the properties of a path and associated X-Y table. Other controls at the bottom involve importing GPS data, and are described in the subsection **Importing GPS Data** (page 22).



Path and X-Y table, each with custom IDs

Path ID: 1001 Table ID: 1001 Start station: 0 m

	Latitude	Longitude	
GPS References			deg

☐ Set references from import file

☑ Looped path Lap length: 2327.9762 m

Path coordinates: X (m), Y (m), S [calculated] (m)

	Global X	Global Y	Station
1	0	0	0.0000
2	8.85	-60.0126	[illegible]
3	9.5251	-64.6151	65.3134
4	10.2247	-69.2276	69.9786
5	10.8244	-73.8541	74.6438
6	11.1633	-78.506	79.3081
7	11.2008	-83.1705	83.9727
8	10.9947	-87.8308	88.6376
9	10.5634	-92.4759	93.3027
10	9.9327	-97.0981	97.9677
11	9.0664	-101.6815	102.6322
12	7.8502	-106.1843	107.2964
13	6.2622	-110.5694	111.9602
14	4.3272	-114.8135	116.6246

Rows: 377

https://atlas.carsim.com

☐ Reumann-Witkam

Figure 13. The Path: X-Y Coordinates screen.

(1) Drop-down list to specify how the Reference Path and X-Y table will be identified. The control offers two options:

1. One option is to add a new path to the model and set the parameter `PATH_ID` automatically to the internal number of the path (e.g., if this is the third path used for a simulation, then `PATH_ID(3)` = 3). It will also define a new X-Y table with the VS Command `DEFINE_XY_TABLES`, and set the ID for the table to the number of the table (e.g., if this is the fifth table defined in the simulation, then `XY_TABLE_ID(5)` is set to 5).
2. The other option shows two fields to specify a custom ID for the path (2) and another for the X-Y table (3). Details for each ID are described below.

(2) Yellow field to specify a custom Path ID (keyword = `SET_IPATH_FOR_ID`). This field is shown when the custom ID option is selected for the drop-down control (1). The value for the ID must be an integer of 999 or greater. The value is checked as described earlier (page 7).

1. If a path has already been defined with the specified ID number, then the information in this dataset is used to change the properties of the existing path.
2. If a path with this ID does not exist, then a new path is created and the `PATH_ID` parameter is set to the value specified in the yellow field.

(3) Yellow field to specify the X-Y Table ID (keyword = `SET_ITAB_XY_FOR_ID`). This field is shown when the Custom ID option is selected for the drop-down control (1). The value for the ID must be an integer of 999 or greater. The ID is handled the same as the Path ID (2), except that it applies to an X-Y Table.

④ Starting station (keyword = `SPATH_START`). This defines the station for the first row in the table. If not specified, the starting station will be zero.

⑤ These two yellow fields may be used to specify the GPS Reference Coordinates, specified in degrees (keywords = `GPS_REF_LAT` and `GPS_REF_LONG`). These are the GPS coordinates at a point in the global coordinate system specified by the parameters `GPS_REF_X` and `GPS_REF_Y`. Unless specified otherwise, the reference point coordinates retain their default values `GPS_REF_X` = 0 and `GPS_REF_Y` = 0.

The VS Solver uses these four parameters to calculate GPS latitude and longitude output variables `GPS_LatA` and `GPSLongA` from global X (east) and global Y (north) coordinates.

When there are multiple paths used in a simulation and each have a specification for the GPS Reference Coordinates, the GPS Reference Coordinates that are specified on the last path processed by the VS Solver are the set used for the simulation.

For more information about how GPS coordinates are handled in VS Math Models, please see the Tech Memo *GPS Coordinates* from the **Help** menu. The use of GPS data to create Reference Paths or segments is described in the subsection **Importing GPS Data** (page 22).

⑥ **Looped path** checkbox (keyword = `OPT_PATH_LOOP`). When this box is checked, the first and last row in the table must have identical X and Y values, and the VS Solvers treat the road descriptions as looping. When this box is checked, the length of the loop is shown ⑥.

⑦ This field is shown only if the **Looped path** box is checked ⑥, in which case it displays the calculated length of the looped path.

⑧ Three-column table of values for X and Y global coordinates of the X-Y table path (keyword = `SEGMENT_XY_TABLE`). Each line requires a value of X followed by a corresponding value of Y. More information about how these are used within the VS Solver is provided in a later section (page 20).

⑨ Each row includes a third variable, S (station), calculated automatically based on X and Y and the starting station ④.

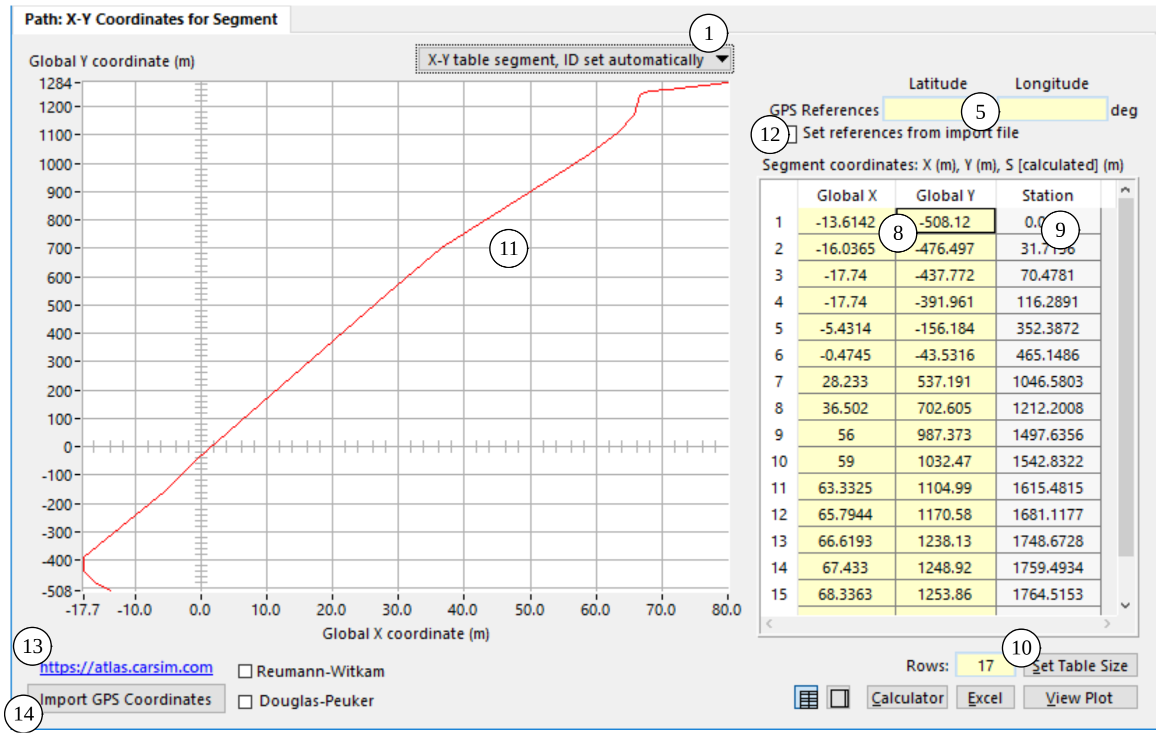
⑩ Use the yellow field to specify the number of rows in the table, then click the adjacent button **Set Table Size** to resize the table.

⑪ Plot of path using X-Y coordinates ⑧. This plot connects the coordinates with straight lines. The VS Browser (GUI) does not perform the custom spline interpolation that the VS Solver performs. Most datasets have far too many points for the difference to be visible. However, if a path has large gaps between some of the points, the spline interpolation might define unintended curves. If the path is associated with a road, you can use the video preview option on the **Road: 3D Surface (All Properties)** screen to check the road shape, as described later.

The remaining controls ⑫ - ⑭ involving importing GPS data to obtain X-Y coordinates, as described in the subsection **Importing GPS Data** (page 22).

Path: X-Y Coordinates for Segment

The **Path: X-Y Coordinates for Segment** library (Figure 14) is less complicated than the **Path: X-Y Coordinates** screen, described above. It has controls related to an X-Y table, but no settings for a path.



GPS References Latitude Longitude deg

☐ Set references from import file

☐ Reumann-Witkam

☐ Douglas-Peuker

Rows: 17

Segment coordinates: X (m), Y (m), S [calculated] (m)

	Global X	Global Y	Station
1	-13.6142	-508.12	0.0
2	-16.0365	-476.497	31.7136
3	-17.74	-437.772	70.4781
4	-17.74	-391.961	116.2891
5	-5.4314	-156.184	352.3872
6	-0.4745	-43.5316	465.1486
7	28.233	537.191	1046.5803
8	36.502	702.605	1212.2008
9	56	987.373	1497.6356
10	59	1032.47	1542.8322
11	63.3325	1104.99	1615.4815
12	65.7944	1170.58	1681.1177
13	66.6193	1238.13	1748.6728
14	67.433	1248.92	1759.4934
15	68.3363	1253.86	1764.5153

Figure 14. The Path: X-Y Coordinates for Segment screen.

Datasets from this library can be linked into a row of a path built using the **Path: Segment Builder** library described earlier (page 10). Each **Path: Segment Builder** dataset can contain up to 100 segments, meaning up to 100 datasets from the **Path: X-Y Coordinates for Segment** library can be used to define a path using the **Path: Segment Builder**.

The X-Y path defined using this **Path: X-Y Coordinates for Segment** library does not specify initial X-Y coordinates, starting station, or starting heading. These are all set automatically to maintain continuity with the previous segment in the path.

Because the X-Y coordinates are all relative to the segment, the same dataset can be used multiple times in a simulation, whether as multiple segments in a single **Path: Segment Builder** dataset or in different **Path: Segment Builder** datasets linked together to create a road network.

The numbered controls shown in Figure 14 were all described previously for the **Path: X-Y Coordinates** screen.

The controls ⑫ - ⑭ involve importing GPS data to obtain X-Y coordinates, and are described in the subsection **Importing GPS Data** (page 22).

The GPS reference fields ⑤ are provided in case GPS data are imported into this screen. Otherwise, the fields should probably be left blank to avoid conflicting with a GPS reference specified on the Path dataset that makes used of the segment.

Calculations Involving Spline X-Y Tables

The library screens described in the previous subsections use X-Y coordinates to define an arbitrary shape (Figure 15). Within the spline segment, station *S* is calculated using straight-line segments; the length of each line segment is added to the previous value of station *S* when going from point to point in a direction for increasing station:

$$S_i = S_{i-1} + \sqrt{(X_i - X_{i-1})^2 + (Y_i - Y_{i-1})^2} \quad \{ i > 1 \}$$

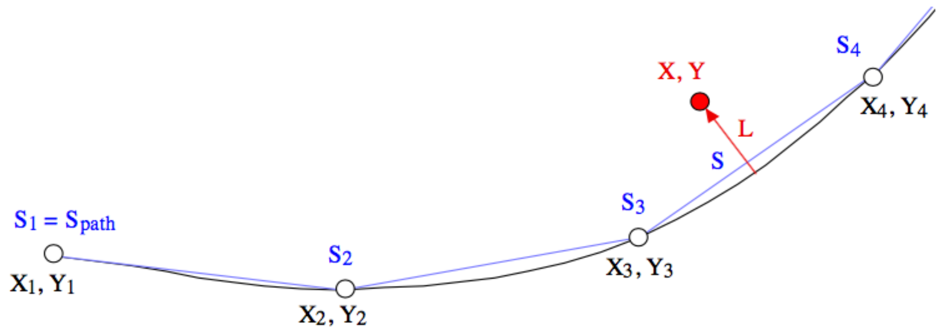


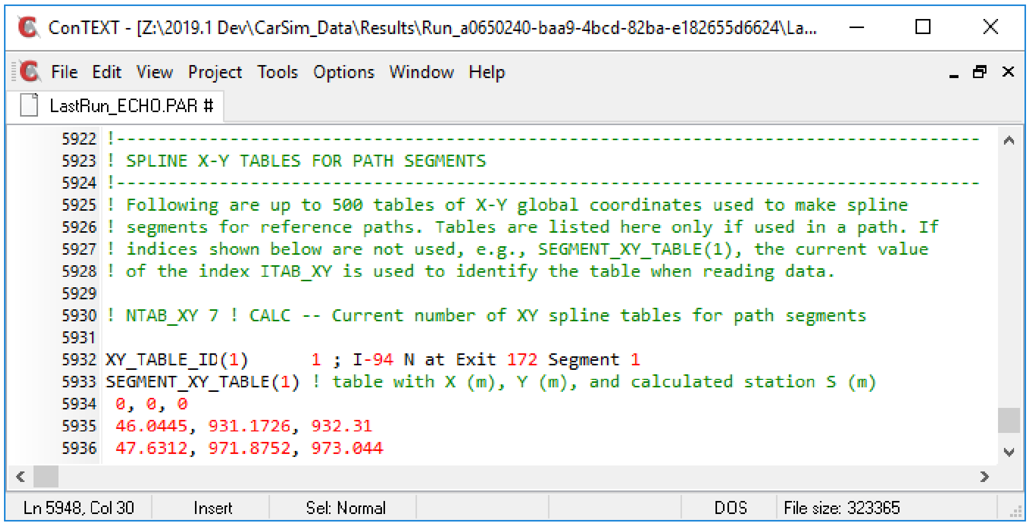
Figure 15. Horizontal geometry for a spline segment defined with X-Y coordinates.

The change in *true station* between two points is slightly larger than the straight-line distance between points because the true station is defined along the curved path.

Although the calculation of station is based on straight-line segments, the reference line between the points is indeed curved, as indicated in Figure 15. At station values between the original data points, there can be an offset between the straight connecting line and the curved path. Internally, the VS Solver programs generate tables of *X* vs. *S* and *Y* vs. *S* and use spline interpolation to calculate *X* and *Y* as continuous functions of station *S* for points on the curved reference line. The spline functions also provide $\partial X/\partial S$ and $\partial Y/\partial S$ as functions of station *S*.

The data for a spline segment might seem similar to those used in other Configurable Functions in a VS Math Model; however, this type of table is unique because the input X and Y coordinates are used to automatically generate corresponding values of a third variable S (station) that is calculated internally. Data for these tables are written in their own section of the Echo file, after all of the Configurable Function data (Figure 16).

The spline tables are not written with the rest of the path information because they are often lengthy when written out. Other types of segments (straight lines, arcs, and clothoids) take just two or three lines (e.g., radius/curvature and angle) and are written within the list of segments for a path (see the Echo file, Figure 10, page 15).


ConTEXT - [Z:\2019.1 Dev\CarSim_Data\Results\Run_a0650240-baa9-4bcd-82ba-e182655d6624\La...
File Edit View Project Tools Options Window Help
LastRun_ECHO.PAR #
Ln 5948, Col 30
Insert
Sel: Normal
DOS
File size: 323365

```
5922 !--------------------------------------------------------------------------------
5923 ! SPLINE X-Y TABLES FOR PATH SEGMENTS
5924 !--------------------------------------------------------------------------------
5925 ! Following are up to 500 tables of X-Y global coordinates used to make spline
5926 ! segments for reference paths. Tables are listed here only if used in a path. If
5927 ! indices shown below are not used, e.g., SEGMENT_XY_TABLE(1), the current value
5928 ! of the index ITAB_XY is used to identify the table when reading data.
5929
5930 ! NTAB_XY 7 ! CALC -- Current number of XY spline tables for path segments
5931
5932 XY_TABLE_ID(1)      1 ; I-94 N at Exit 172 Segment 1
5933 SEGMENT_XY_TABLE(1) ! table with X (m), Y (m), and calculated station S (m)
5934  0, 0, 0
5935  46.0445, 931.1726, 932.31
5936  47.6312, 971.8752, 973.044
```

Figure 16. Echo file showing start of X-Y Table for a path segment.

As noted before, each table has a parameter specifying a user ID number, such as `XY_TABLE_ID(1)` (line 5932, Figure 16). Notice that the description was set by the Browser to match the title of the dataset where the table was defined: `I-94 N at Exit 172 Segment 1`.

Importing GPS Data

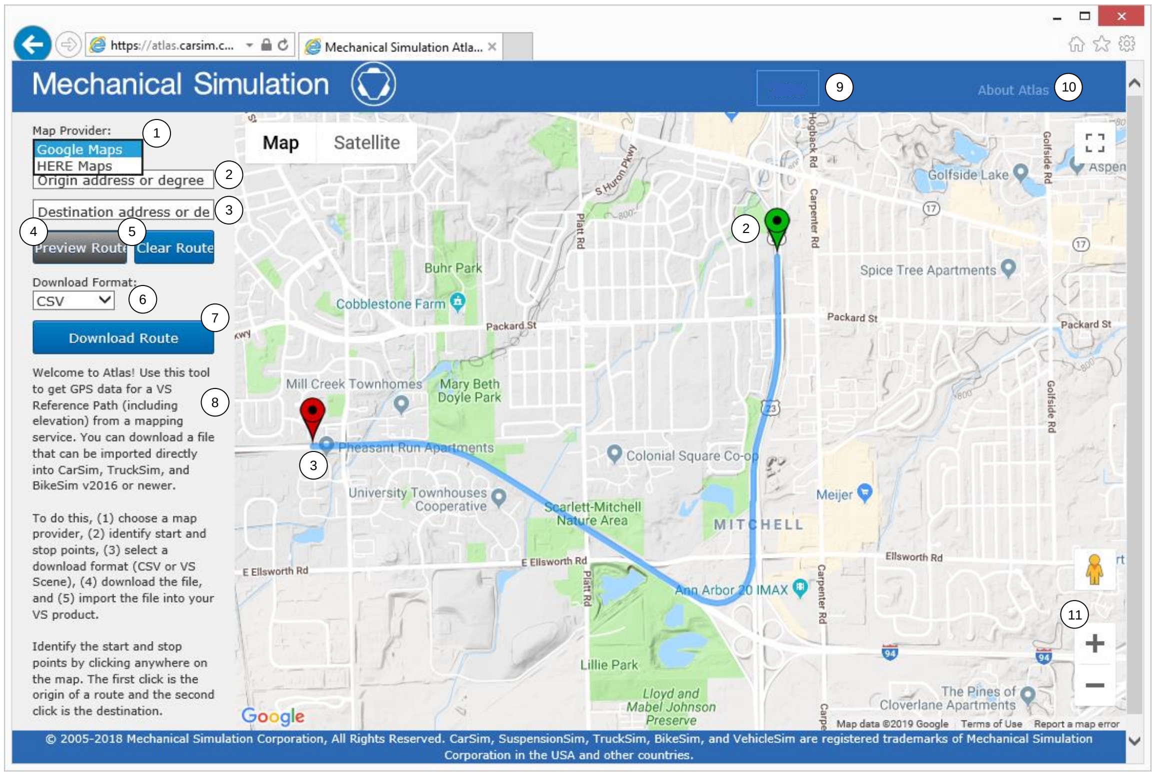
The screens for X-Y path data can work with latitude and longitude coordinates using the decimal degree format. On the X-Y screens (Figure 13 and Figure 14) there is a button named **Import GPS Coordinates** (14). Clicking this button will open a file-select dialog to specify a CSV file to import. The format of the CSV file contains decimal degree latitude and longitude values separated by a comma, e.g.:

```
41.66395,-83.55521
41.66394,-83.55553
41.66391,-83.55713
41.66386,-83.5591
41.66383,-83.56057
```

Once a properly formatted CSV file is specified, the geodetic coordinates will be converted to Cartesian coordinates. GPS coordinates are calculated using a local linear relationship between global X and GPS longitude, and global Y and GPS latitude. The relationship is set using VS parameters `GPS_REF_LAT`, `GPS_REF_LONG`, `GPS_REF_X`, and `GPS_REF_Y`, as described in the document *GPS Coordinates* (from the menu **Help** > **Technical Memos**).

Atlas Web Application

Mechanical Simulation maintains a web application called Atlas (https://atlas.carsim.com), serving as a source of GPS data that are intended to be converted to X-Y paths in BikeSim, CarSim, and TruckSim (Figure 17). Atlas can be accessed either by typing the web address directly into an internet browser or by clicking the link in the VS Browser (Figure 13 and Figure 14, (13)).



Welcome to Atlas! Use this tool to get GPS data for a VS Reference Path (including elevation) from a mapping service. You can download a file that can be imported directly into CarSim, TruckSim, and BikeSim v2016 or newer.

To do this, (1) choose a map provider, (2) identify start and stop points, (3) select a download format (CSV or VS Scene), (4) download the file, and (5) import the file into your VS product.

Identify the start and stop points by clicking anywhere on the map. The first click is the origin of a route and the second click is the destination.

© 2005-2018 Mechanical Simulation Corporation, All Rights Reserved. CarSim, SuspensionSim, TruckSim, BikeSim, and VehicleSim are registered trademarks of Mechanical Simulation Corporation in the USA and other countries.

Figure 17. Atlas, a web application for generating GPS data.

Once Atlas launches, there are two choices of map provider: Google or HERE. Which map provider to choose depends on a variety of factors: each of these companies have their own methods of collecting road data and turning that into a map, meaning that once a set of origin and destination coordinates have been selected in Atlas, the route that is generated using Google may not be the same as the route generated using HERE. It is worth noting that the map data from either of these sources is of the same quality that one can expect from driving directions.

Other considerations include the location in which you are working; one or both of these map data source providers may not be available in your location. In such a case, the GPS coordinates will have to be generated by some other means. Once the GPS data is in a CSV file in decimal degree format, the **Import GPS Coordinates** button in the VS Browser can be used to import that data and create an X-Y path.

1. Drop-down menu to specify the map data source provider. Options are Google and HERE.

2. Field to specify the origin (first point) of the path using either an address or GPS coordinates. As an alternative to typing data in the field, you can also click in the field, then click in the map. This will set the origin point for the path. If desired, you can drag and reposition the origin point in the map.

3. Field to specify the destination (final point) of the path using either an address or GPS coordinates. As an alternative to typing data into the field, you can also click in the field, then click in the map. This will set the destination point for the path. If desired, you can drag and reposition the destination point in the map.

④ Button to **Preview Route** once the origin and destination coordinates have been selected. Clicking this button will draw the route on the map.

⑤ Button to **Clear Route**, after the route has been previewed using ④. If the route drawn using **Preview Route** doesn't yield the desired result, you can try repositioning the origin and/or destination markers. If the route is still not of interest, clicking the button **Clear Route** will allow you to clear what has been drawn and try again.

⑥ Drop-down menu to specify the format of the downloaded data. Options are CSV and VsScene. The CSV file format is the original format introduced with Atlas with BikeSim / CarSim / TruckSim version 2016.0, and the resulting file can be imported into one of several Path screens described earlier. The VsScene option is a newer format that is intended to be imported into the datasets in the library **Scene: External Import**. The **Scene: External Import** library screens can also import the CSV file format.

Note GPS elevation data is not included in the CSV file.

⑦ Button to **Download Route**. Clicking this button will create a file in the selected format ⑥ based on the selected route, and download that file to your computer. After saving the file, it can be imported into one of several Path screens in the VS Browser.

⑧ Overview of instructions for using Atlas.

⑨ Single Sign-On account information. Access to Atlas is part of the Single Sign-On policy implemented by Mechanical Simulation and includes access to the User Section of the website (www.carsim.com). If you are not currently signed into your account on www.carsim.com when Atlas is accessed, you will be prompted to log-in using the same credentials as would be used to download software installers.

⑩ **About Atlas**. Clicking this button will bring up a window with tabs describing Release Notes, Historical Changes, and some documentation describing the workflow of Atlas. The documentation can also be accessed from the BikeSim / CarSim / TruckSim Help drop-down menu, *Atlas GPS Tool* (**Help** > **Paths, Road Surfaces, and Scenes**).

⑪ Buttons to zoom-in/zoom-out on the map. To pan the viewing area, hold left- or right-click while dragging the mouse. Zooming may also be accomplished with a mouse's scroll wheel.

Smoothing Path Data

Sometimes X-Y path data can be noisy and produce unacceptable road geometry; GPS data is particularly susceptible to this. To reduce the noise, it is possible to smooth your X-Y path data using two different algorithms (Figure 18).

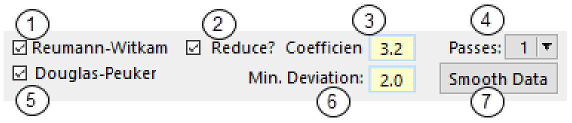


Figure 18. Smoothing options common to X-Y Table screens.

① Toggle this checkbox to enable/disable the Reumann-Witkam smoothing algorithm. This is an incremental perpendicular distance algorithm, which distinguishes itself by measuring test points using a line from a key point to future points, as opposed to previous and future points.

② Toggle this checkbox to put the Reumann-Witkam smoothing algorithm in Reductive Mode. When checked, any test point outside the smoothing distance will be removed from the table. When unchecked, the algorithm will instead move the point to lay along the smoothed path. This control is only visible when ① is checked.

③ This value describes how strict the Reumann-Witkam algorithm should be when considering test points. Valid values are within the range 0 to 10. A value of 0 will apply no smoothing, and a value of 10 will result in a straight line. Determining the proper value for this field can require some trial-and-error. This control is only visible when ① is checked.

④ This drop-down control determines how many successive passes the Reumann-Witkam algorithm will perform on the data. This control is only visible when ① is checked.

⑤ Toggle this checkbox to enable/disable the Douglass-Peuker smoothing algorithm. This is a polyline simplification algorithm which locates the point along the curve, and whose removal would minimize the Hausdorff distance between the original curve and the new curve without the point. This is done recursively until either the resultant curve is a straight line, or the next candidate point is within a minimum distance of the resultant curve. This option will not be available if the X-Y Table describes a looped path.

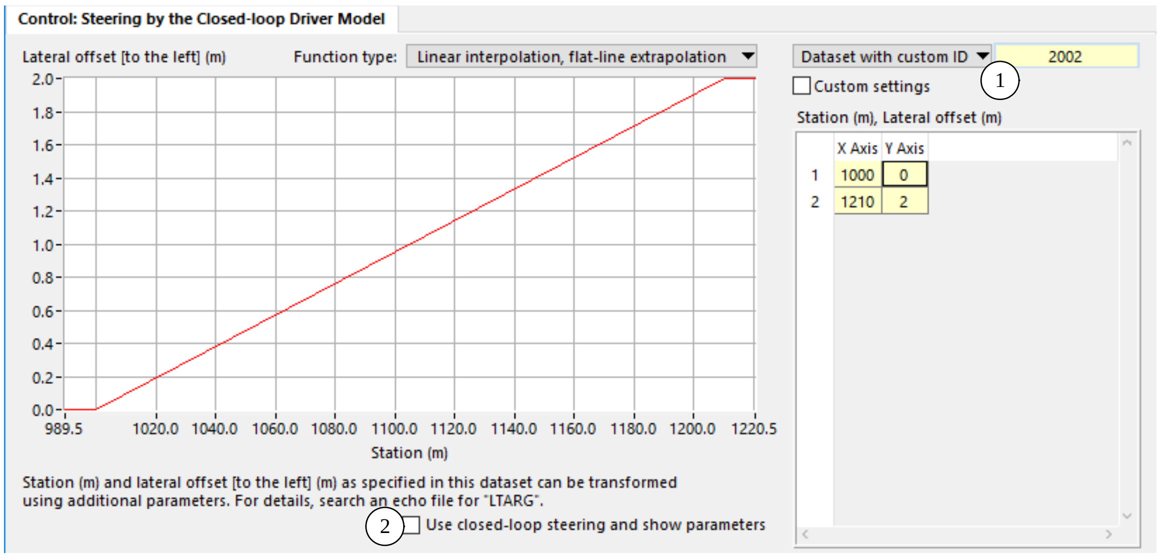
⑥ This value determines the minimum distance to consider candidate points within the Douglass-Peuker smoothing algorithm. Valid values must be positive. Determining the proper value for this field can require some trial-and-error. This control is only visible when ⑤ is checked.

⑦ Clicking this button causes the GUI to apply the smoothing algorithms as configured to the X-Y Table. If the smoothing is not acceptable, and edits must be made to the configuration, first undo the already applied smoothing and then apply the new smoothing options. Successive attempts to smooth data, particularly high-density data, may result in undesirable results.

The LTARG Configurable Function

BikeSim, CarSim, and TruckSim include a Configurable Function `LTARG` that calculates a lateral offset L as a function of station S. This function was originally provided to define a target path for the Closed-Loop Path-Following steering controller, for lane changes and other targets that were based on a Reference Path. More recently, the `LTARG` function is also used to control the motions of moving objects, relative to a Reference Path, representing traffic vehicles, lane markings, ADAS sensor targets, and other features of interest.

New `LTARG` datasets are defined in the VS Browser with the library screen **Control: Steering by the Closed-loop Driver Model** (Figure 19). The screen has an option for setting a custom ID to the dataset ①, and a checkbox to show driver control parameters ②. If the **Driver Model** dataset exists for the sole purpose of adding an `LTARG` dataset, uncheck the box ② (as shown) to avoid interacting with driver model settings made elsewhere.



Function type: Linear interpolation, flat-line extrapolation

Dataset with custom ID: 2002

☐ Custom settings

Station (m), Lateral offset (m)

	X Axis	Y Axis
1	1000	0
2	1210	2

Station (m) and lateral offset [to the left] (m) as specified in this dataset can be transformed using additional parameters. For details, search an echo file for "LTARG".

☐ Use closed-loop steering and show parameters

Figure 19. LTARG datasets are created using the Driver Model screen.

When specifying information for the function (a constant, a table of S-L values, etc.) from the GUI screen, the Parsfile written by the VS Browser sets the index parameter `ILTARG` for either a new dataset, or for an existing dataset matching the custom ID (e.g., 2002) ①.

The `LTARG` dataset shown in Figure 19 is one of three that were used in a simulation. All three are listed in the Echo file (Figure 20).

Note that three `LTARG` datasets are shown. The comments indicate that up to 500 can be used (line 3156). Of the three, the first has `LTARG_ID(1)` = 1 (the default when the ID is set automatically), the second has `LTARG(2)` = 2001, and the third has `LTARG(3)` = 2002.

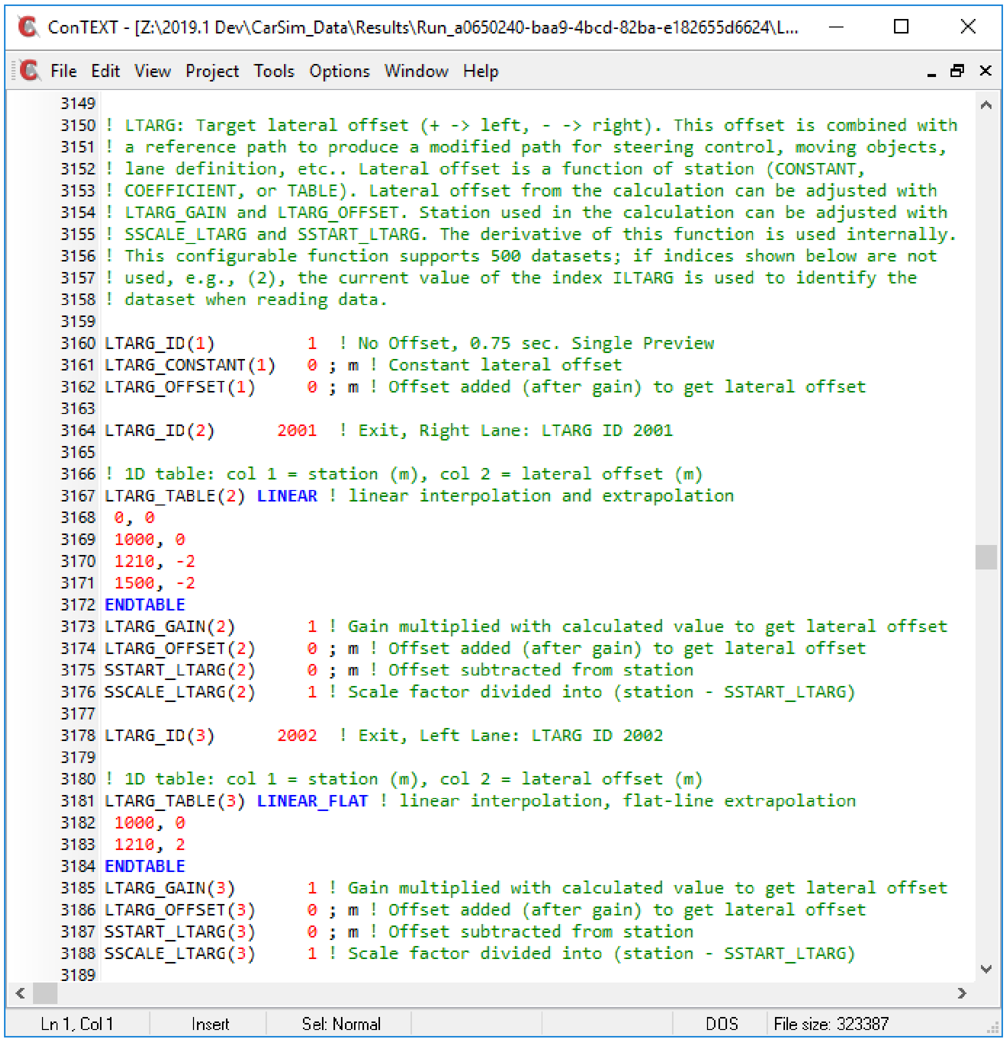
The `LTARG` dataset used by the driver / rider model is specified with the parameter `LTARG_ID_DM` (line 839, Figure 21). Datasets used for moving objects are specified with the object parameter `LTARG_ID_OBJ(`*i*`)`, where *i* is the object number.

The total number of active `LTARG` datasets is the parameter `N_LTARG`, shown in the Echo file section for driver/rider model (line 837, Figure 21). It is normally adjusted automatically by commands written by the VS Browser in `LTARG` dataset Parsfiles, which are applied when the Parsfiles are read by the VS Solver.

Road Surface Physical Properties

A surface performs several functions in BikeSim, CarSim, and TruckSim:

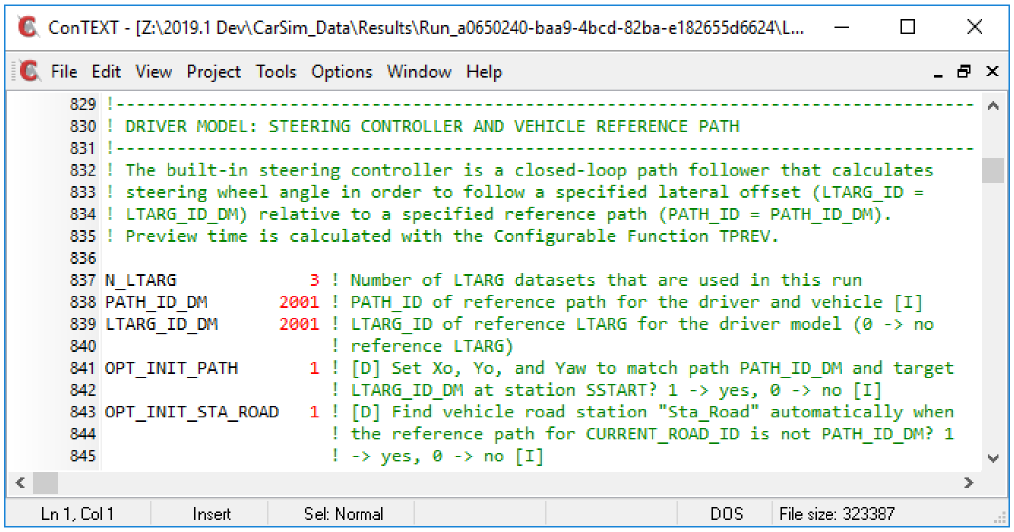
1. It defines the ground geometry and friction where the tires contact the ground in the vehicle model.
2. It controls the elevation and 3D orientation (pitch and roll) of moving objects that may have been added to the simulation using the VS Solver.
3. It is used to define a road axis system under the vehicle, using the aerodynamic reference point on the sprung mass of the vehicle model.



ConTEXT - [Z:\2019.1 Dev\CarSim_Data\Results\Run_a0650240-baa9-4bcd-82ba-e182655d6624\L...
File Edit View Project Tools Options Window Help

```
! LTARG: Target lateral offset (+ -> left, - -> right). This offset is combined with
! a reference path to produce a modified path for steering control, moving objects,
! lane definition, etc.. Lateral offset is a function of station (CONSTANT,
! COEFFICIENT, or TABLE). Lateral offset from the calculation can be adjusted with
! LTARG_GAIN and LTARG_OFFSET. Station used in the calculation can be adjusted with
! SSCALE_LTARG and SSTART_LTARG. The derivative of this function is used internally.
! This configurable function supports 500 datasets; if indices shown below are not
! used, e.g., (2), the current value of the index ILTARG is used to identify the
! dataset when reading data.

LTARG_ID(1)          1  ! No Offset, 0.75 sec. Single Preview
LTARG_CONSTANT(1)    0 ; m ! Constant lateral offset
LTARG_OFFSET(1)      0 ; m ! Offset added (after gain) to get lateral offset

LTARG_ID(2)       2001  ! Exit, Right Lane: LTARG ID 2001

! 1D table: col 1 = station (m), col 2 = lateral offset (m)
LTARG_TABLE(2) LINEAR ! linear interpolation and extrapolation
 0, 0
 1000, 0
 1210, -2
 1500, -2
ENDTABLE
LTARG_GAIN(2)        1 ! Gain multiplied with calculated value to get lateral offset
LTARG_OFFSET(2)      0 ; m ! Offset added (after gain) to get lateral offset
SSTART_LTARG(2)      0 ; m ! Offset subtracted from station
SSCALE_LTARG(2)      1 ! Scale factor divided into (station - SSTART_LTARG)

LTARG_ID(3)       2002  ! Exit, Left Lane: LTARG ID 2002

! 1D table: col 1 = station (m), col 2 = lateral offset (m)
LTARG_TABLE(3) LINEAR_FLAT ! linear interpolation, flat-line extrapolation
 1000, 0
 1210, 2
ENDTABLE
LTARG_GAIN(3)        1 ! Gain multiplied with calculated value to get lateral offset
LTARG_OFFSET(3)      0 ; m ! Offset added (after gain) to get lateral offset
SSTART_LTARG(3)      0 ; m ! Offset subtracted from station
SSCALE_LTARG(3)      1 ! Scale factor divided into (station - SSTART_LTARG)

```

Figure 20. Section of Echo file showing LTARG datasets.



ConTEXT - [Z:\2019.1 Dev\CarSim_Data\Results\Run_a0650240-baa9-4bcd-82ba-e182655d6624\L...
File Edit View Project Tools Options Window Help

```
!------------------------------------------------------------------------------------
! DRIVER MODEL: STEERING CONTROLLER AND VEHICLE REFERENCE PATH
!------------------------------------------------------------------------------------
! The built-in steering controller is a closed-loop path follower that calculates
! steering wheel angle in order to follow a specified lateral offset (LTARG_ID =
! LTARG_ID_DM) relative to a specified reference path (PATH_ID = PATH_ID_DM).
! Preview time is calculated with the Configurable Function TPREV.

N_LTARG              3 ! Number of LTARG datasets that are used in this run
PATH_ID_DM        2001 ! PATH_ID of reference path for the driver and vehicle [I]
LTARG_ID_DM       2001 ! LTARG_ID of reference LTARG for the driver model (0 -> no
                       ! reference LTARG)
OPT_INIT_PATH        1 ! [D] Set Xo, Yo, and Yaw to match path PATH_ID_DM and target
                       ! LTARG_ID_DM at station SSTART? 1 -> yes, 0 -> no [I]
OPT_INIT_STA_ROAD    1 ! [D] Find vehicle road station "Sta_Road" automatically when
                       ! the reference path for CURRENT_ROAD_ID is not PATH_ID_DM? 1
                       ! -> yes, 0 -> no [I]
```

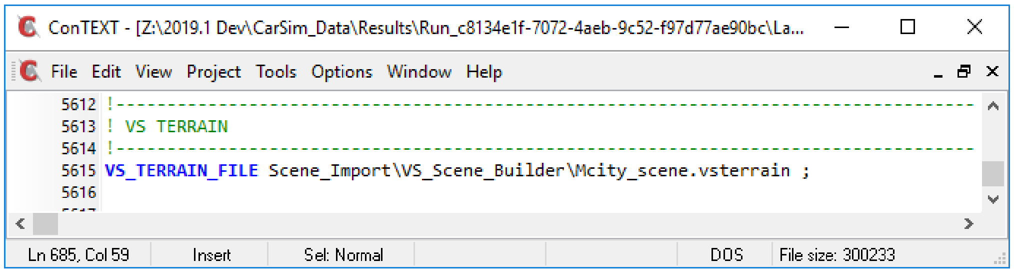
Figure 21. Echo file showing the Driver Model: Steering Controller section.

These functions may be handled by either a set of VS Roads, or a single VS Terrain.

VS Roads are also used by the VS Browser to generate 3D animator shapes automatically for visualization purposes.

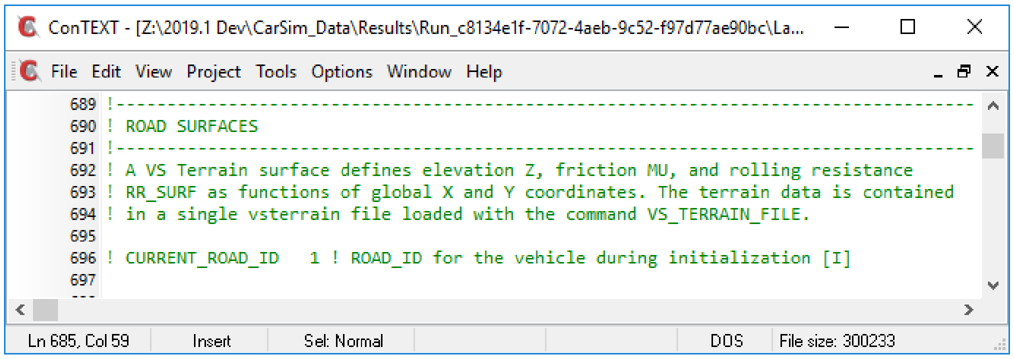
The VS Terrain Option

As noted earlier, VS Solvers support two different approaches for describing ground geometry. With the VS Terrain option, a single terrain file contains all information about the ground surface that is needed by the VS Solver. The file is accessed using the command `VS_TERRAIN_FILE`, usually applied when using a **Scene: External Import** dataset. If this command was used, it is shown near the end of the Echo file (Figure 22). When a `vsterrain` file is used, VS Roads are disabled, as indicated by comments in the Road Surfaces section of the Echo file (Figure 23).


ConTEXT - [Z:\2019.1 Dev\CarSim_Data\Results\Run_c8134e1f-7072-4aeb-9c52-f97d77ae90bc\La...
File Edit View Project Tools Options Window Help
Ln 685, Col 59
Insert
Sel: Normal
DOS
File size: 300233

```
!-------------------------------------------------------------------------------
! VS TERRAIN
!-------------------------------------------------------------------------------
VS_TERRAIN_FILE Scene_Import\VS_Scene_Builder\Mcity_scene.vsterrain ;
```

Figure 22. The VS_TERRAIN_FILE option.


ConTEXT - [Z:\2019.1 Dev\CarSim_Data\Results\Run_c8134e1f-7072-4aeb-9c52-f97d77ae90bc\La...
File Edit View Project Tools Options Window Help
Ln 685, Col 59
Insert
Sel: Normal
DOS
File size: 300233

```
!-------------------------------------------------------------------------------
! ROAD SURFACES
!-------------------------------------------------------------------------------
! A VS Terrain surface defines elevation Z, friction MU, and rolling resistance
! RR_SURF as functions of global X and Y coordinates. The terrain data is contained
! in a single vsterrain file loaded with the command VS_TERRAIN_FILE.

! CURRENT_ROAD_ID   1 ! ROAD_ID for the vehicle during initialization [I]
```

Figure 23. The Road Surfaces section of the Echo file is empty when a vsterrain file is used.

As indicated in the Echo file, the parameter `CURRENT_ROAD_ID` is set to a value of 1 and cannot be changed. The ID of 1 may also be used to specify that a moving object is positioned vertically using the VS Terrain surface via the parameter `ROAD_ID_OBJ`.

As noted earlier, the VS Terrain option is described in the *VS Terrain* and *VS Scene Builder* tech memos and the *Scene External Import* screen document. The remainder of this section along with the following section on connecting road surfaces apply mostly for VS Roads. However, the last sections (Initial Vehicle Locations, Roughness Profiles, Vehicle-Road Axis Systems, and VS Commands) apply for both VS Roads and VS Terrain. Also, the **Road: 3D Surfaces (All Properties)** screen is used to link a VS Terrain file to the VS Solver (page 33).

Be aware that when the Road screen is used to link to a **Scene: External Import** dataset that is linked to a VS Terrain file, many of the controls on the screen are hidden because they are not applicable (Figure 24).

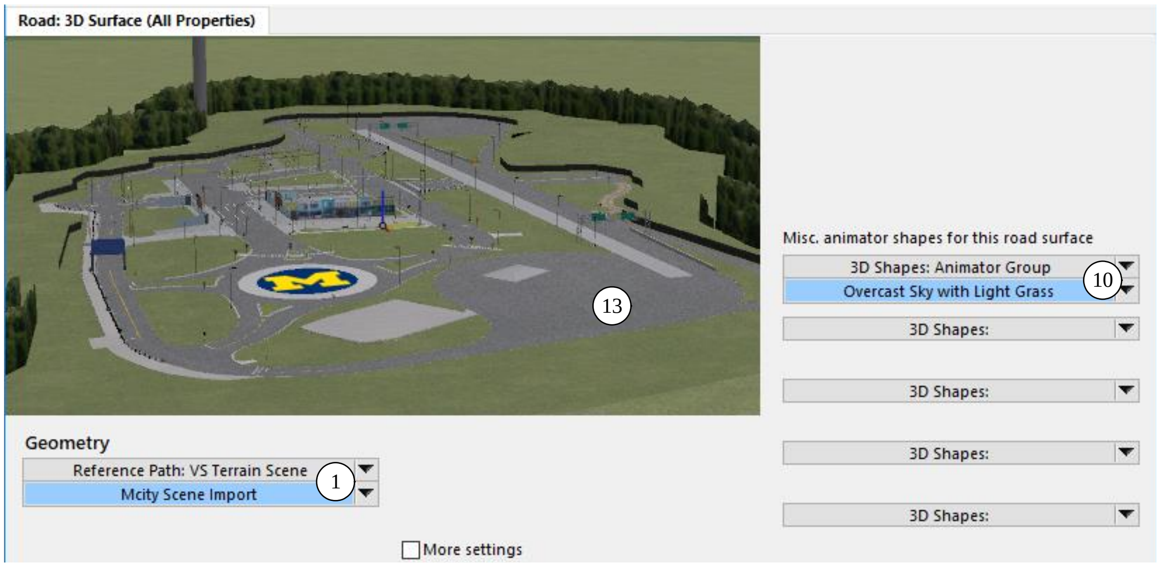

Road: 3D Surface (All Properties)
13
Misc. animator shapes for this road surface
3D Shapes: Animator Group
Overcast Sky with Light Grass
10
3D Shapes:
3D Shapes:
3D Shapes:
3D Shapes:
Geometry
Reference Path: VS Terrain Scene
Mcity Scene Import
1
More settings

Figure 24. The Road: 3D Surfaces (All Properties) screen is simplified when using VS Terrain.

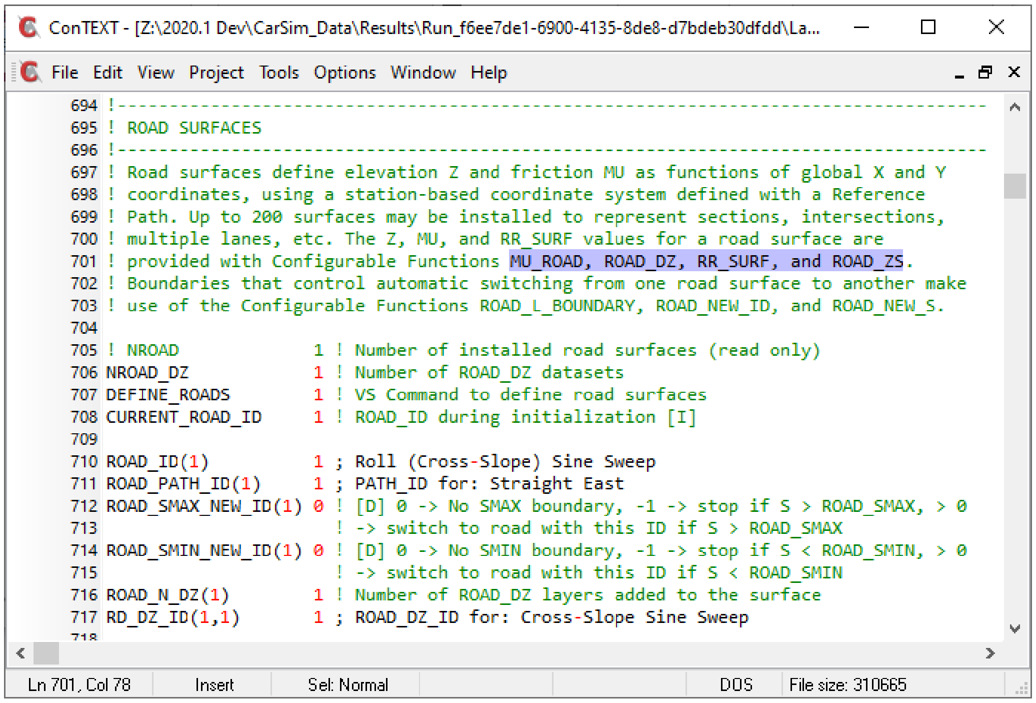
The VS Roads Option

Every VS simulation includes a VS Terrain or at least one VS Road. If one is not defined by linking to a **Road: 3D Surface (All Properties)** dataset, then a VS Road is automatically created during initialization of the VS Solver. The default road added in this case uses a path that is also created automatically as a straight line oriented with a heading of zero, such that S = global X and L = global Y. The elevation of the default surface is zero everywhere, the friction coefficient is set to –1.0 (this is the VehicleSim convention to use tire test data directly without adjustment for road friction), and the rolling resistance coefficient for the surface is zero.

Parameters for all VS Roads defined for a simulation are listed in the Road Surfaces section of the Echo file (Figure 25). This section follows the Reference Paths section of the Echo file shown earlier (Figure 3, page 6).

Along with the parameters shown in this section of the Echo file, each road surface has at least three associated datasets for Configurable Functions.

1. `ROAD_ZS` defines the elevation Z along the Reference Path as a function of station S.
2. `MU_ROAD` defines a friction coefficient Mu as a function of both S and L.
3. `RR_SURF` defines a rolling resistance coefficient as a function of both S and L.


ConTEXT - [Z:\2020.1 Dev\CarSim_Data\Results\Run_f6ee7de1-6900-4135-8de8-d7bdeb30dfdd\La...
File Edit View Project Tools Options Window Help
Ln 701, Col 78
Insert
Sel: Normal
DOS
File size: 310665

```
694 !--------------------------------------------------------------------------------
695 ! ROAD SURFACES
696 !--------------------------------------------------------------------------------
697 ! Road surfaces define elevation Z and friction MU as functions of global X and Y
698 ! coordinates, using a station-based coordinate system defined with a Reference
699 ! Path. Up to 200 surfaces may be installed to represent sections, intersections,
700 ! multiple lanes, etc. The Z, MU, and RR_SURF values for a road surface are
701 ! provided with Configurable Functions MU_ROAD, ROAD_DZ, RR_SURF, and ROAD_ZS.
702 ! Boundaries that control automatic switching from one road surface to another make
703 ! use of the Configurable Functions ROAD_L_BOUNDARY, ROAD_NEW_ID, and ROAD_NEW_S.
704
705 ! NROAD              1 ! Number of installed road surfaces (read only)
706 NROAD_DZ             1 ! Number of ROAD_DZ datasets
707 DEFINE_ROADS         1 ! VS Command to define road surfaces
708 CURRENT_ROAD_ID      1 ! ROAD_ID during initialization [I]
709
710 ROAD_ID(1)           1 ; Roll (Cross-Slope) Sine Sweep
711 ROAD_PATH_ID(1)      1 ; PATH_ID for: Straight East
712 ROAD_SMAX_NEW_ID(1) 0 ! [D] 0 -> No SMAX boundary, -1 -> stop if S > ROAD_SMAX, > 0
713                        ! -> switch to road with this ID if S > ROAD_SMAX
714 ROAD_SMIN_NEW_ID(1) 0 ! [D] 0 -> No SMIN boundary, -1 -> stop if S < ROAD_SMIN, > 0
715                        ! -> switch to road with this ID if S < ROAD_SMIN
716 ROAD_N_DZ(1)         1 ! Number of ROAD_DZ layers added to the surface
717 RD_DZ_ID(1,1)        1 ; ROAD_DZ_ID for: Cross-Slope Sine Sweep
```

Figure 25. Road Surfaces section of an Echo file.

Additional links may exist to datasets for the Configurable Function `ROAD_DZ`, which defines a layer of incremental elevation DZ that is a function of S and L, and is added to the road surface elevation. Other links are used to define optional boundary conditions for the road surface, as will be described in a later section **Connecting Road Surfaces** (page 51).

The VS Road used to initialize the vehicle model is specified with the vehicle parameter `CURRENT_ROAD_ID`, which is assigned to the `ROAD_ID` parameter for the road of interest (line 708, Figure 25). If the selected road has boundaries, those boundaries are considered when calculating the initial conditions of the vehicle. For example, different tires might be located on different road surfaces based on the boundary specifications, as described later (page 51). More information about the initialization is provided in a later section (page 63).

VS Roads can be used to control the elevation and 3D orientation (pitch and roll) of moving objects. Each object has a parameter `ROAD_ID_OBJ` that can be assigned to the `ROAD_ID` value of an existing road. If the road has boundaries, the road ID for each object is updated as the simulation proceeds, as described in a later section **Connecting Road Surfaces** (page 51).

VS Roads sometimes defined strictly for cosmetic reasons: to generate detailed 3D shape files with textures mapped to follow curves, show lanes, grass, etc. Be aware that tires and moving objects don’t know anything about animator cosmetics. If the elevation, friction, and rolling resistance are uniform, then the vehicle model might not need the road. When using the **Road: 3D Surface (All Properties)** to create animator shapes and textures, you can reuse the same road ID multiple times. Only the last dataset loaded will be listed in the Echo file generated for the simulation, but VS Visualizer will show all the shapes.

If multiple roads have different Reference Paths, but all are flat and have the same constant friction coefficient, then an efficient way to define the simulation environment is to make use of the multiple paths associated with each road, but specify one more road surface that is straight, flat, and has consistent friction. For example, if the straight flat road has ID number 1000, then set `CURRENT_ROAD_ID` to 1000. On the other hand, set the driver model path `PATH_ID_DM` to the path of interest and possibly change the path during the simulation using VS Events, while leaving `CURRENT_ROAD_ID` with the initial value. The same approach can be used for moving objects. For example, set the road ID for each object to 1000 (or `CURRENT_ROAD_ID`), but set the path ID parameter `PATH_ID_OBJ` for a specific path, and possibly change it during the run by using VS Events.

The Road: 3D Surface (All Properties) Screen

A VS Road Surface defines elevation based on an S-L-Z coordinate system.

Elevation (Z_{road}) is defined as a function of S and L as:

$$Z_{road} = Z_S\,(S, id) + dZ\,(L, S, i) + dZ\,(L, S, j) + dZ\,(L, S, k) + \ldots \tag{1}$$

where:

- Z_S is a function of *S* and a road index number *id* (id = 1, …)
- *dZ* is an optional function of *S*, *L*, and a dataset ID (*i, j, k, …*) that adds a layer of incremental elevation to the road.

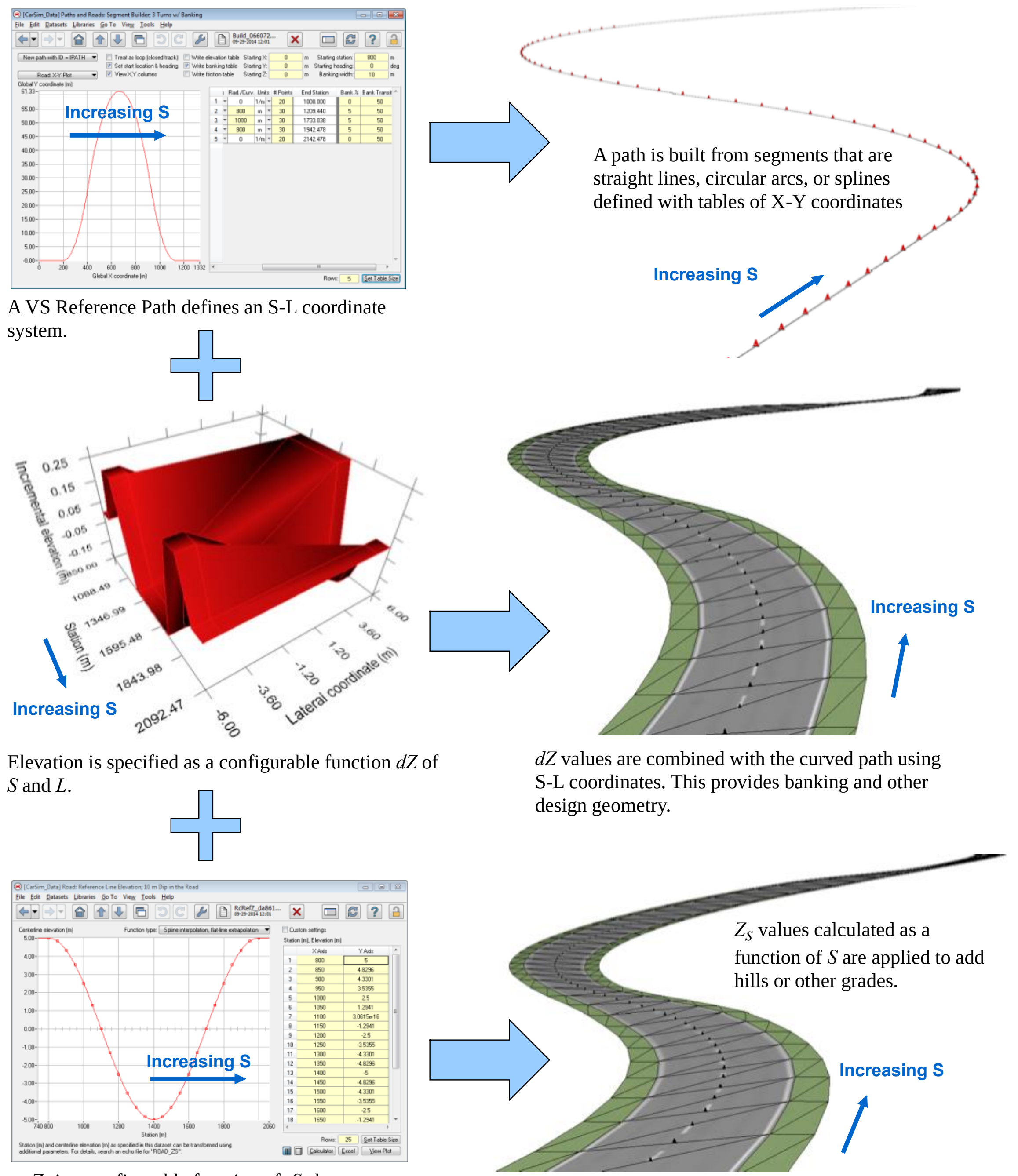
Each road *id* has a related dataset for the Configurable Function `ROAD_ZS` that is used to calculate the Z_S component. By default, a road has zero incremental layers, but up to 200 can be added. If a layer is added, then a dataset is specified with a user ID for the Configurable Function `ROAD_DZ`.

Figure 26 illustrates the method implied by Eq. (1) for building a 3D road surface from tabular data based on S and L coordinates. Images on the left side of the figure show screens from the VS Browser used to enter and/or visualize the road data. Images on the right-hand side (generated from VS Visualizer) show the road geometry as it is built from each component.

First, a Reference Path is provided (see the upper-left part of the figure). In this example, the road turns to the left, then to the right, and then to the left to resume the original heading angle. In the Echo file, the Path used for the road is identified with the parameter `ROAD_PATH_ID`. All roads in BikeSim, CarSim, and TruckSim must have an X-Y Reference Path; if one is not linked, the default behavior is a straight path that is oriented East-West such that increasing station is heading East.

Second, banking of turns involves changes in elevation away from the reference line. The middle-left part of the figure shows a 3D map of elevation in terms of *S* and *L*. The geometry here is simple; there are three banking angles for the three curves, separated with linear transitions. The road shown in the middle-right part of the figure shows the effect of taking the *dZ* shape and distorting it to follow the curved path, with *S* and *L* defined from the curved reference line. In the Echo file, this 3D shape is defined with a specified dataset for the Configurable Function `ROAD_DZ`.

Third, the lower-left part of the figure shows a screen for setting Z_S as a function of *S* alone, with the road dropping 10 m in elevation about halfway through the second curve. In this case, the Configurable Function was set to use spline interpolation within the range of data provided, with flat-line extrapolation. In the Echo file, this component of elevation is defined with the dataset for the Configurable Function `ROAD_ZS`.



New path with ID = IPATH; Road: X-Y Plot; Treat as loop (closed track) [ ]; Set start location & heading [✓]; View X,Y columns [✓]; Write elevation table [ ]; Write banking table [✓]; Write friction table [ ]; Starting X: 0 m; Starting Y: 0 m; Starting Z: 0 m; Starting station: 800 m; Starting heading: 0 deg; Banking width: 10 m

	Rad./Curv.	Units	# Points	End Station	Bank %	Bank Transit
1	0	1/m	20	1000.000	0	50
2	800	m	30	1209.440	5	50
3	1000	m	30	1733.038	5	50
4	800	m	30	1942.478	5	50
5	0	1/m	20	2142.478	0	50

Rows: 5

A VS Reference Path defines an S-L coordinate system.

A path is built from segments that are straight lines, circular arcs, or splines defined with tables of X-Y coordinates

Elevation is specified as a configurable function *dZ* of *S* and *L*.

dZ values are combined with the curved path using S-L coordinates. This provides banking and other design geometry.

Z_S values calculated as a function of *S* are applied to add hills or other grades.

Function type: Spline interpolation, flat-line extrapolation; Custom settings [ ]

Station (m), Elevation (m)

	X Axis	Y Axis
1	800	5
2	850	4.8296
3	900	4.3301
4	950	3.5355
5	1000	2.5
6	1050	1.2941
7	1100	3.0615e-16
8	1150	-1.2941
9	1200	-2.5
10	1250	-3.5355
11	1300	-4.3301
12	1350	-4.8296
13	1400	-5
14	1450	-4.8296
15	1500	-4.3301
16	1550	-3.5355
17	1600	-2.5
18	1650	-1.2941

Rows: 25

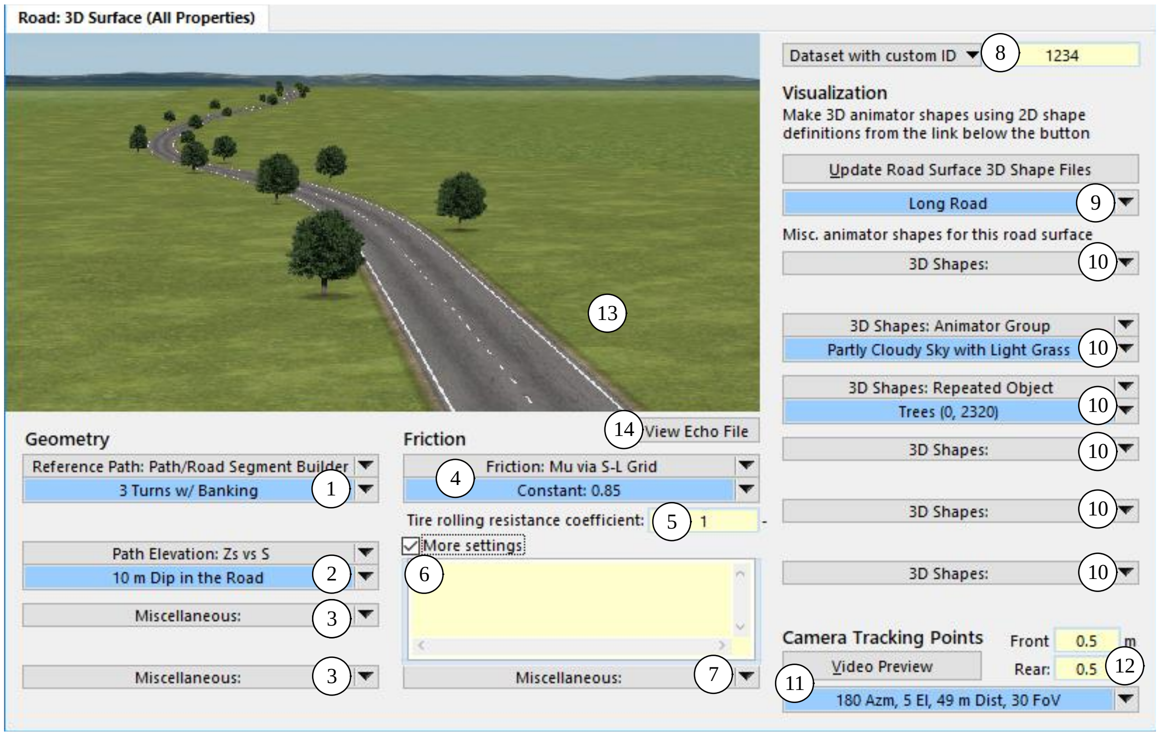
Station (m) and centerline elevation (m) as specified in this dataset can be transformed using additional parameters. For details, search an echo file for "ROAD_ZS".

Z_S is a configurable function of *S* alone.

Figure 26. The 3D road description is built from Configurable Functions.

The view of the road in the lower-right part of the figure shows the combined result of the horizontal curve geometry, the off-center elevation due to banking, and the 10-m dip in the middle of the section.

Use the **Road: 3D Surface (All Properties)** screen to assemble these components to define complete 3D road surface, along with associated shapes for visualization (Figure 27).


Road: 3D Surface (All Properties)

Dataset with custom ID: 1234

Visualization

Make 3D animator shapes using 2D shape definitions from the link below the button

Update Road Surface 3D Shape Files

Long Road

Misc. animator shapes for this road surface

3D Shapes:

3D Shapes: Animator Group — Partly Cloudy Sky with Light Grass

3D Shapes: Repeated Object — Trees (0, 2320)

3D Shapes:

3D Shapes:

3D Shapes:

Camera Tracking Points — Front: 0.5 m, Rear: 0.5

Video Preview

180 Azm, 5 El, 49 m Dist, 30 FoV

View Echo File

Geometry

Reference Path: Path/Road Segment Builder — 3 Turns w/ Banking

Path Elevation: Zs vs S — 10 m Dip in the Road

Miscellaneous:

Miscellaneous:

Friction

Friction: Mu via S-L Grid — Constant: 0.85

Tire rolling resistance coefficient: 1 -

☑ More settings

Miscellaneous:

Figure 27. The Road: 3D Surface (All Properties) screen.

Note Please see *Road Surface Visualization* (**Help** > **Paths, Road Surfaces, and Scenes**) for a focus on how the screen is used to assemble data to visualize the 3D road surface.

The controls in the lower-left part of the screen define the physical properties of the surface (1 - 7). The ID for the road is controlled with the drop-down control 8 and adjacent yellow field, a button 14 shows an Echo file listing all of the path and road data for the assembly, and the remaining controls define how the road surface will appear in VS Visualizer.

Geometry

Every VS Road has an associated Reference Path, specified with the `ROAD_PATH_ID` parameter, friction as specified with the `MU_ROAD` Configurable Function, and elevation of the Reference Path as specified with the `ROAD_ZS` function. The surface may also include up to 200 incremental elevation layers, specified with the `ROAD_DZ` function.

Alternatively, this screen may be used to link to a VS Terrain dataset using the first blue link 1.

1 Link to a dataset that is either for a Reference Path that defines the S-L coordinate system for the road, or for a VS Terrain file that typically includes animation assets. The VS Browser contains six libraries that can be used here, listed in Table 1.

Table 1. Libraries that provide horizontal geometry.

Library Name	**Road Properties Included**	**Skip Links**
Path: Segment Builder	Reference path	
Path: X-Y Coordinates	Reference path	
Road: X-Y-Z Coordinates of Reference Line	Reference path, Z of path	②
Road: X-Y-Z Coordinates of Edges	Reference path, Z of path, dZ_1	②
Path/Road: Segment Builder (Legacy)	Ref. path, Z of path, dZ_1, friction	②④
Scene: External Import	Reference path, possibly Z of path	
	VS Terrain file	See Figure 24

Each of these library screens can provide the Reference Path to define the S-L coordinate system for the road. Each provides a different combination of additional properties, simplifying the use of common forms of road data that are sometimes available. The last column in the table identifies the links on this screen that are typically not needed, because the information is already included in the selected horizontal geometry dataset.

The three Path libraries listed in the table were described earlier in the section on Reference Paths (page 5). The two Road libraries (X-Y-Z coordinates) are described later in this section.

The **Scene: External Import** library supports multiple types of data, including a Reference Path only, or a Reference Path plus Z as a function of station, or a link to a VS Terrain file. If the linked dataset is for a VS Terrain file, then many of the controls are not applicable and are hidden (Figure 24, page 29).

The **Scene: External Import** library is described in the document *Scene External Import* (**Help** > **Paths, Road Surfaces, and Scenes**).

② Link to a **Road: Reference Line Elevation** dataset. This specifies the vertical geometry of the road Reference Path using with a Configurable Function that calculates Z as a function of station S. The root keyword for the table is `ROAD_ZS`.

This link is not shown if the horizontal geometry link ① points to a dataset in which the X-Y-Z coordinates are specified (Table 1) or in which a VS Terrain file is specified.

If the link for horizontal geometry goes to a dataset from the **Path/Road: Segment Builder (Legacy)** library or the **Scene: External Import** library, the elevation link may still be shown. Datasets from these libraries may optionally specify elevation information; if this is the case, and a link is made here to a **Road: Reference Line Elevation** dataset, then the specifications from the **Elevation** dataset will override those from the **Builder** or **External Import** dataset.

③ Miscellaneous links, typically used to add incremental elevation layers to the road surface with datasets from the **Road: Off-Center Elevation Map, S-L Grid** or **Road: Off-Center Elevation Map, Variable Width** libraries that make use of the `ROAD_DZ` Configurable Function (see details in a following section, page 38).

Note Prior to version 2017, road surfaces all had two layers defined with built-in `ROAD_DZ` Configurable Function datasets. In newer versions, the layers are optional, and the default number of layers is zero. The number of allowable layers has been increased from 2 to 200.

Layers are added to a VS Road by linking to a `ROAD_DZ` dataset. If no links are made, then no layers are added. The datasets for the `ROAD_DZ` function are created as needed; if no links are made to `ROAD_DZ` dataset, then none are added to the math model.

The maximum number of layers allowed is 200, and the current number of installed datasets is assigned to the variable `NROAD_DZ`, which is listed in the Echo file (see line 698, Figure 25, page 30). To include more than three layers (③ and ⑦), use one of the Miscellaneous links to connect with a Generic Links dataset that can in turn include many links to more `ROAD_DZ` datasets.

Alternatively, these links can be used to include animation datasets that are related to the road surface.

④ Link to a **Road: Friction Map, S-L Grid** or **Road: Friction Map, Variable Width** dataset. This specifies friction as a function of longitudinal and lateral position on the road with a Configurable Function. Of the two libraries, the former supports a constant, a linear coefficient, a 1D interpolation involving only S, or a 2D interpolation involving S and L; the latter library supports 2D variable-width interpolation. The root keyword for all cases is `MU_ROAD`.

This link is still shown even if the link for horizontal geometry ① goes to a dataset from the **Paths/Road: Segment Builder (Legacy)** library. The **Segment Builder** dataset can optionally specify friction information; if it does, and a link is made here to a **Friction** dataset, then the specifications from the **Friction** dataset will override those from the **Builder** dataset.

⑤ Rolling resistance coefficient for the VS Road (keyword = `RR_SURF_CONSTANT`). Tire rolling resistance is calculated with the equation:

MyRR = R*Fz*RR_surf*(RR_c + RR_v*Vx)

where MyRR is a moment about the spin axis, R is the instant radius, Fz is vertical load, RR_surf is the coefficient specified in this yellow field, Vx is forward speed, and RR_c and RR_v are tire properties.

The surface coefficient is represented in the math model with a 2D Configurable Function `RR_SURF` that varies with S and L. If the simulation conditions involve the vehicle leaving a pavement, or encountering other differences in surface conditions, one of the miscellaneous blue links (③ and ⑦) may be made to a dataset from the **Road: Rolling Resistance Map, S-L Grid** or **Road: Rolling Resistance Map, Variable Width** libraries. If linking to a dataset one of these libraries, leave the yellow field ⑤ blank.

Custom Settings

⑥ Checkbox for **Custom settings** with a Miscellaneous data field.

The miscellaneous data field can contain any text that would be recognized by the VS Solver, including VS Commands to add new variables.

7 **Miscellaneous** link. This can be made to many of the VS Browser data screens. Some popular links are VS Commands, extra animation settings, a ROAD_DZ layer, and **Road Boundaries** datasets (ideal if the boundaries only involve the current road; if a boundaries dataset involves another road or a set of roads, the link should not be made here).

Road ID Number

8 Drop-down control to specify an ID number for this road.

Each VS Road has an ID number that is used to specify the road as the surface in contact with the tires or optional moving objects. It becomes important only when a simulation involves multiple road surfaces. If the simulation involves a single road surface, the best option is usually **New dataset; set ID automatically**. This option adds a new road and sets the ID number automatically the internal index for the road. For example, if this dataset is used to define the first road in a scenario, then the VS Solver will set the parameter ROAD_ID(1) to 1. If it is the third road added, then the parameter ROAD_ID(3) would be set to 3.

If the second option is selected (Figure 28), a data field is shown to specify an ID number for this dataset. The ID number is used to identify a road surface for initializing the vehicle, specifying elevation for optional moving objects, and specifying boundary conditions for other road surfaces.

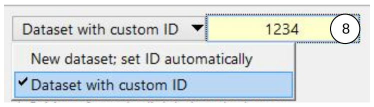


Figure 28. Options for setting a road ID number.

The custom ID should have a value of 999 or greater, and is handled the same as the Path custom IDs described earlier (page 7), except they are applied to road surfaces instead of paths.

Each Road dataset includes the statement:

```
CURRENT_ROAD_ID = ROAD_ID
```

which assigns the road to the vehicle for initialization. If a simulation includes multiple roads, the last one that is added is used to initialize the vehicle. If the simulation includes multiple road surfaces, it is a good idea to specify CURRENT_ROAD_ID in a miscellaneous field that is read by the VS Solver after all of the roads have been read.

Note The ROAD_ID parameter does not exist for VS Terrain, and the CURRENT_ROAD_ID parameter is automatically set to 1 when VS Terrain is used.

Road Surface Visualization

The remaining user controls concern the appearance of the road in VS Visualizer. This screen makes use of the VS Road Calculator that was described earlier.

VS Road Calculator supports the full set of VS Road and VS Path commands and data types that are available in the VS Solvers for the vehicle math models. As a VS Solver, it reads from Parsfiles, works with the VS API, and can generate text Echo files. However, unlike the other VS Solvers, it contains no vehicle math model, and is not used to run time-domain simulations. Instead, the VS Browser employs it to calculate X, Y, and Z coordinates for 3D shapes that are used by VS Visualizer to show the 3D road surface.

⑨ **Update Road Surface 3D Shape Files** button and link to a dataset from the **Road: Animator Surface Shapes** library.

The linked **Road: Animator Surface Shapes** dataset contains cosmetic information that is used by VS Visualizer to show the road surface. This library is described in a separate **Help** document: **Road Surface Visualization**.

When a link exists to a **Road: Animator Surface Shapes** dataset, that information is used with VS Road Calculator to generate 3D shapes to animate the road surface. The shape files are re-written whenever a change is made to the **Road Animator Surface Shapes** dataset. However, if you edit one of the linked datasets on the **Road: 3D Surface (All Properties)** screen, those changes will not trigger an automatic rewriting of the animator shape descriptions. If you edit a linked dataset that defines the road geometry (①, ②, ③, ⑦) or appearance (⑨, ⑩), but do not make any changes to the data on the **Road: Animator Surface Shapes** screen, then you should click the **Update Road Surface 3D Shape Files** button to ensure the road shape files are rewritten.

If there is no link to a **Road: Animator Surface Shapes** dataset, then shapes will not be generated and the button to update the shape files is dimmed. Examples of when not to link to the **Road: Animator Surface Shapes** library include the use of `.osg` animation files or the **Scene: External Import** screen. The **Scene: External Import** screen may contain a link to a `vsscene` file, generated by the VS Scene Builder (**Tools > VS Scene Builder**…). The VS Scene Builder includes animation assets defined by the tiles used to create that scene.

⑩ Links to Miscellaneous animation data. Use these links to add features to the road that are affected by elevation geometry, such as houses, trees, etc. You can link to the **Road: Animator Repeated Object** to add multiple copies of an object using global or road coordinates.

The **Road: Animator Repeated Object** library is described in the document *Road Surface Visualization* (**Help -> Paths, Road Surfaces, and Scenes)**.

The **Single Moving Object (Custom)** library can be used to create a detectable polygonal shape that takes advantage of station-lateral coordinates when linked from the **Road: 3D Surface (All Properties)** screen.

⑪ **Video Preview** button and link to dataset from the **Animator: Camera Setup** library. Click to view the 3D surface as defined by the links on this screen for the road geometry plus the animator shape settings. The camera settings are set with the link immediately under this button. This button is dimmed if there is no linked **Animator: Camera Setup** dataset.

⑫ Positions of road reference camera tracking points for preview animations (keywords = `L_CAMERA_FRONT` and `L_CAMERA_REAR`). The VS Solvers for all VehicleSim vehicle models can be set to calculate output variables associated with two tracking points: one leading the vehicle and one following the vehicle. The variables have names such as `X_CamF`, `Y_CamF`, `X_CamR`, etc. and are used to set up tracking camera views for the animator.

The two values set here are distances in front of, and behind, the station of the vehicle on the driver Reference Path (specified with the parameter `PATH_ID_DM`.

Note The output variables used to be built in. However, in some conditions where the driver model is not used, such as driving simulators, attempts to calculate the variables can cause numerical problems and an error message. The variables and supporting parameter are no longer built-in. However, they may be installed with the command `INSTALL_CAMERA_OUTPUTS`.

(13) Custom graphic. Left-click on the picture to pull down a menu to provide an image that will visually identify this dataset. The image is typically created with these steps:

1. View the road with VS Visualizer by clicking the video preview button (11).
2. Adjust the view in VS Visualizer and copy it to the Windows clipboard with Ctrl+C or with a right-click popup menu.
3. Left-click in the graphic area (14) and select the option **Paste picture from the clipboard**.

See the document *VS Browser (GUI and Database)* for more options on making a custom graphic (**Help** > **Reference Manuals**).

Road Calculator

(14) **View Echo File** button. As noted above, the 3D shapes that are made to visualize the road surface are calculated with VS Road Calculator. The calculator reads all the path and elevation data available from the current dataset. As with any VS Solver, it generates an Echo file that shows all the road parameters, the Reference Path, and associated Configurable Function datasets. Use this button the view the Echo file.

Road Elevation Screens

The elevation (global Z coordinate) of a road surface is uniquely defined as a function of coordinates *S* and *L*. Here is equation 1, repeated from page 31.

$$Z_{road} = Z_S\,(S, id) + dZ\,(L, S, i) + dZ\,(L, S, j) + dZ\,(L, S, k) + \ldots \qquad (1)$$

where Z_S is a function of *S* and a road index number *id* ($id = 1, 2, \ldots$), and *dZ* is an optional function of *S*, *L*, and a dataset ID (*i, j, k, …*) that adds a layer of incremental elevation to the road.

Each road *id* has a related dataset for the Configurable Function `ROAD_ZS` that is used to calculate the Z_S component. The layers represented as *dZ* in the equation are provided by datasets for the `ROAD_DZ` function.

Three screens are available for specifying elevation independently of the horizontal geometry. They are available for use if the information was not already embedded in a horizontal geometry dataset from one of the libraries described later in this section.

1. **Road: Reference Line Elevation**. Use this screen to specify the elevation of the Reference Path (Z_S) as a function of S using the Configurable Function `ROAD_ZS`.

2. **Road: Off-Center Elevation Map, S-L Grid**. Use this screen to specify the off-center elevation changes (*dZ*) as functions of S-L coordinates using the Configurable Function `ROAD_DZ` when a grid of S-L coordinates is sufficient.

3. **Road: Off-Center Elevation Map, Variable Width**. Use this screen to specify the off-center elevation changes (*dZ*) as functions of S-L coordinates for roads whose widths vary or have elevation changes that are not necessarily parallel or perpendicular to the reference line of the road.

The elevation maps use Configurable Function controls described in the document *VS Browser (GUI and Database)* (**Help** > **Reference Manuals**).

Road: Reference Line Elevation Screen

Use this screen to specify a component of elevation as a function of station (Z_S) for the Reference Path used for the road. This screen has a standard VS Browser interface to a 1D Configurable Function. The root keyword for the function is `ROAD_ZS`, and it uses the same system index `IROAD` as is used for road surface parameters. For example, tabular data for road 3 would appear in the Echo file with the keyword `ROAD_ZS_TABLE(3)`.

If no link is made from the Road 3D screen, the default setting for `ROAD_ZS` is used: a constant value of zero.

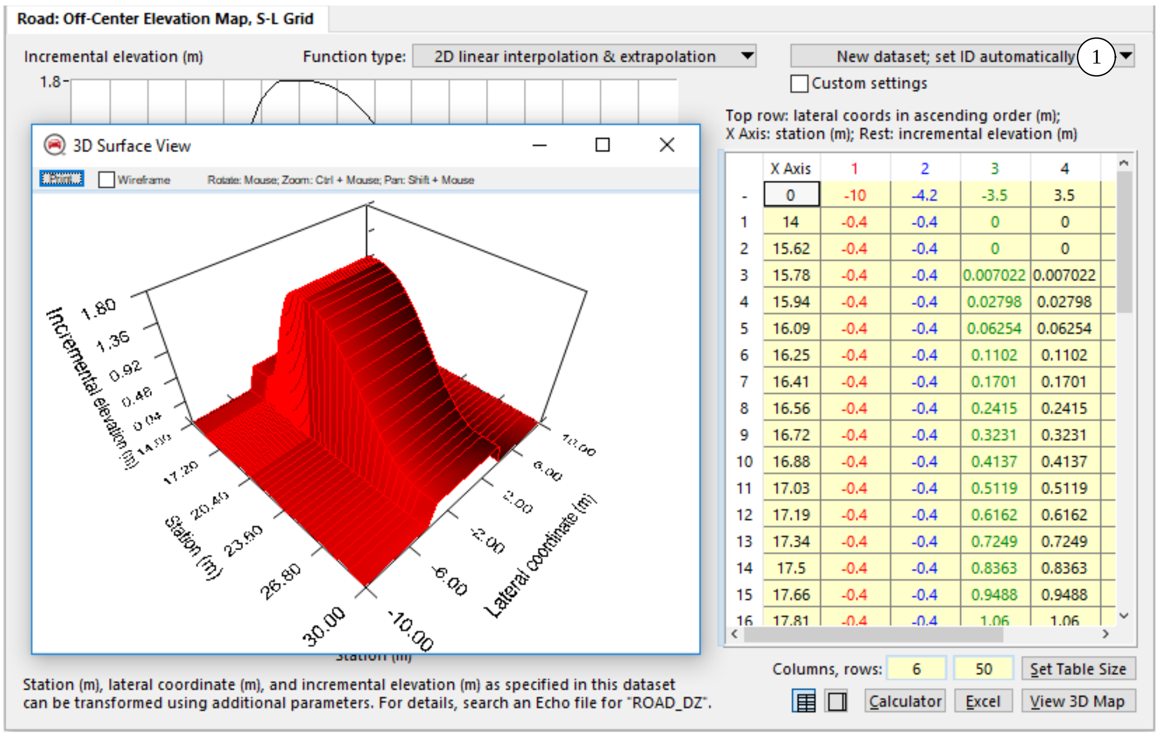
Road: Off-Center Elevation Map, S-L Grid Screen

Use this screen to specify an incremental contribution *dZ* using the Configurable Function `ROAD_DZ`. This screen has a standard VS Browser interface to a Configurable Function (Figure 29), with one additional setting ① to set an ID for the dataset. The `ROAD_DZ` function can be configured to provide a constant, a linear coefficient, a 1D interpolation involving only S, or a 2D interpolation involving S and L.

Note When used with the 2D interpolation, the S-L grid is a curved rectangular grid (CRG). The CRG method concept has been widely applied in representing 3D surfaces in math models.

In recent years, when the acronym CRG is applied to road surfaces, the term is sometimes assumed to apply to a representation developed by Daimler called OpenCRG. OpenCRG is part of another open file format OpenDRIVE, released in 2009. Both are now maintained by VIRES GmbH, a simulation technology company.

An OpenCRG file cannot be read directly by a VS Math Model.


Road: Off-Center Elevation Map, S-L Grid

Incremental elevation (m)

Function type: 2D linear interpolation & extrapolation

New dataset; set ID automatically ⓵

☐ Custom settings

Top row: lateral coords in ascending order (m);
X Axis: station (m); Rest: incremental elevation (m)

	X Axis	1	2	3	4
-	0	-10	-4.2	-3.5	3.5
1	14	-0.4	-0.4	0	0
2	15.62	-0.4	-0.4	0	0
3	15.78	-0.4	-0.4	0.007022	0.007022
4	15.94	-0.4	-0.4	0.02798	0.02798
5	16.09	-0.4	-0.4	0.06254	0.06254
6	16.25	-0.4	-0.4	0.1102	0.1102
7	16.41	-0.4	-0.4	0.1701	0.1701
8	16.56	-0.4	-0.4	0.2415	0.2415
9	16.72	-0.4	-0.4	0.3231	0.3231
10	16.88	-0.4	-0.4	0.4137	0.4137
11	17.03	-0.4	-0.4	0.5119	0.5119
12	17.19	-0.4	-0.4	0.6162	0.6162
13	17.34	-0.4	-0.4	0.7249	0.7249
14	17.5	-0.4	-0.4	0.8363	0.8363
15	17.66	-0.4	-0.4	0.9488	0.9488
16	17.81	-0.4	-0.4	1.06	1.06

Columns, rows: 6 50

Station (m), lateral coordinate (m), and incremental elevation (m) as specified in this dataset can be transformed using additional parameters. For details, search an Echo file for "ROAD_DZ".

Figure 29. *Road Off-Center Elevation, S-L Grid.*

As with some of the other path and road-related screens, this one also has drop-down control ⓵ with an option to define an ID number for the dataset. As with other ID settings, the control has options to add a new table and assign the ID automatically, or to specify an ID and add a new table only if a table does not already exist with the specified ID.

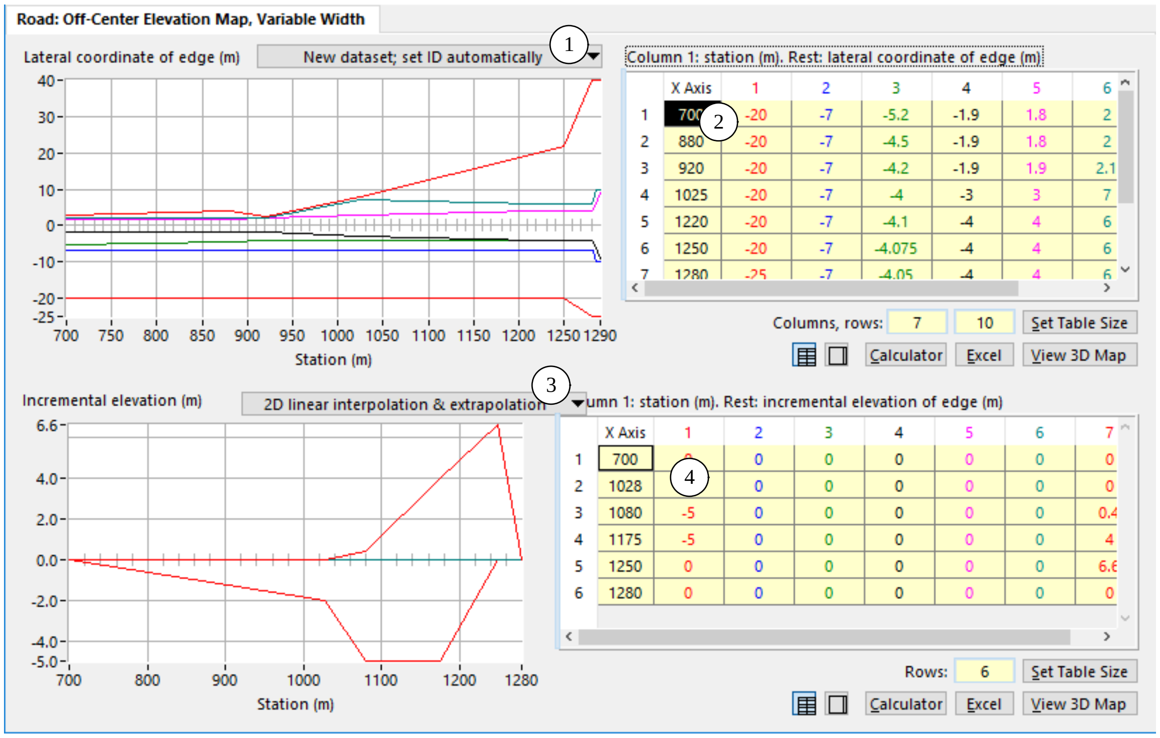
The custom ID should have a value of 999 or greater, and is handled the same as the Path custom IDs described earlier (page 6), except they are applied to `ROAD_DZ` functions instead of paths.

Road: Off-Center Elevation Map, Variable Width

The variable-width library screen defines an elevation component based on an edge number (1, 2, 3…), which in turn is defined as a variable function of station (Figure 30).

This screen also has an option ⓵ for creating a new table and setting the ID automatically, or specifying the ID and make a new table if one is not found with the specified ID.

This screen has familiar controls for Configurable Functions. However, in this case, there are two tables that are closely coupled: one defines the lateral position of an arbitrary lane ID (1, 2, 3…) ⓶ and the other defines incremental elevation *dZ* as a function of S and lane ID ⓸. The table with Z as a function of S and ID can be set for linear or step interpolation ⓷. For example, values of elevation provided in column 3 of a table of input data can correspond to values of *L* that in turn depend on *S*.



Lateral coordinate of edge (m): New dataset; set ID automatically

Column 1: station (m). Rest: lateral coordinate of edge (m)

	X Axis	1	2	3	4	5	6
1	700	-20	-7	-5.2	-1.9	1.8	2
2	880	-20	-7	-4.5	-1.9	1.8	2
3	920	-20	-7	-4.2	-1.9	1.9	2.1
4	1025	-20	-7	-4	-3	3	7
5	1220	-20	-7	-4.1	-4	4	6
6	1250	-20	-7	-4.075	-4	4	6
7	1280	-25	-7	-4.05	-4	4	6

Columns, rows: 7 10

Incremental elevation (m): 2D linear interpolation & extrapolation

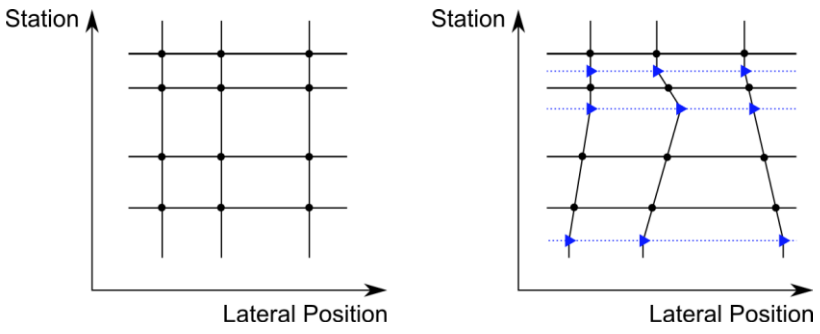
umn 1: station (m). Rest: incremental elevation of edge (m)

	X Axis	1	2	3	4	5	6	7
1	700	0	0	0	0	0	0	0
2	1028	[illegible]	0	0	0	0	0	0
3	1080	-5	0	0	0	0	0	0.4
4	1175	-5	0	0	0	0	0	4
5	1250	0	0	0	0	0	0	6.6
6	1280	0	0	0	0	0	0	0

Rows: 6

Figure 30. Road: Off-Center Elevation Map, Variable Width screen.

With the variable width elevation map, the lateral position of the columns of values can change as a function of station. This allows the user to define, for instance, a road that narrows or widens. This could also be used to define a road with ruts that wander laterally or add a bump to the road that is not perpendicular to the centerline. Figure 31a shows a top view of how the elevation points may be sampled with a rectangular fixed-width map, and Figure 31b shows a variable-width map.

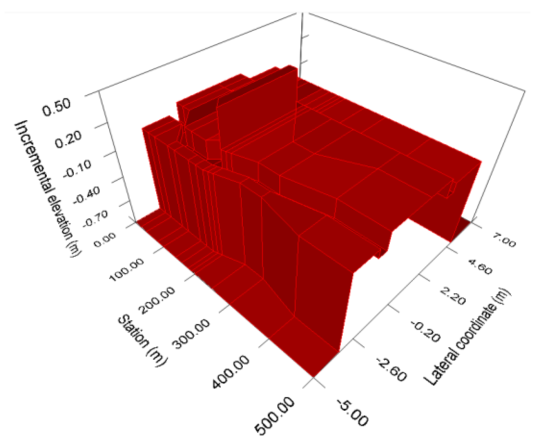


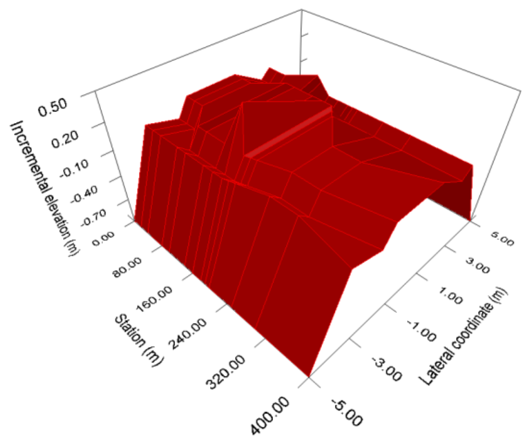
a. fixed-width map *b. variable-width map*

Figure 31. Top view of data maps for fixed- and variable-width off-center maps.

In both schemes, there are three columns of data (edges), which specify the elevation of the road at certain points. In the variable width scheme on the right, the edges do not have to be parallel to the design path. The blue triangles in the figure represent a separate table, which defines the lateral positions of each edge along station. The elevation points are then strung along these lines.

Additionally, there are two types of interpolation methods available for the elevation: step interpolation and linear interpolation. Figure 32 shows an example of the same dataset with these two different interpolation options. In the case of the step interpolation (Figure 32a), the terrain has no roll or pitch at any point, because elevation changes are instantaneous. The 2D linear interpolation connects adjacent data points with straight lines (Figure 32b).





a. step (no interpolation) *b. linear interpolation*

Figure 32. Variable-width interpolation types: step and linear interpolations.

This option can greatly reduce the amount of data needed to define simple geometric features that are not parallel or perpendicular to the road Reference Path. For example, the dataset shown in Figure 30 defines a highway exit lane that widens and has traffic directions on paint (Figure 33).

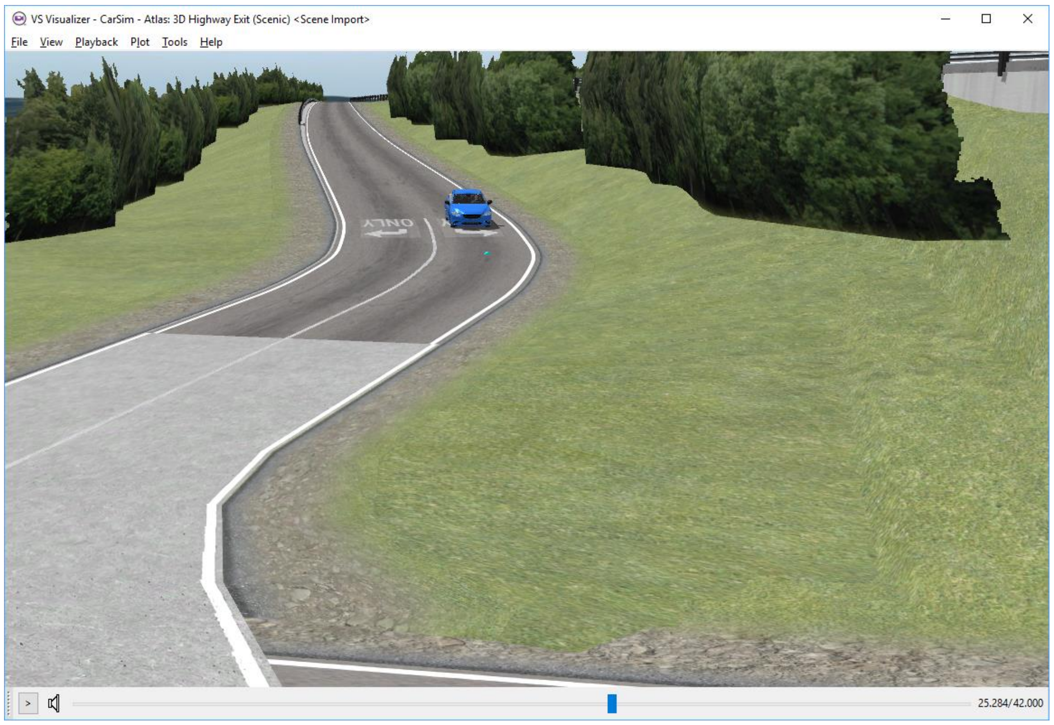

VS Visualizer - CarSim - Atlas: 3D Highway Exit (Scenic) <Scene Import>
File
View
Playback
Plot
Tools
Help
ONLY
25.284/42.000

Figure 33. Variable-width tables simplify the descriptions of exit ramps.

Vertical elevation of the road surface should be specified within 0.1 millimeter to provide a realistic description of typical highway surface that is travelled at highway speeds. However, some databases with road geometry use global coordinate systems that provide unique locations with X-Y-Z coordinates for Earth. Even though the range for a specific table might only cover a few hundred meters (e.g., 43,679,500 m to 43,679,700 m), the extra leading digits can result in round-off error of a millimeter, a centimeter, or more. This round-off is equivalent to adding a component to the data that appears to the VS Math Model as extra roughness.

The VS Browsers were changed May 2010 to always write tabular data with 12 significant digits, essentially eliminating this potential source of error. However, if you are working with older versions, be aware that the browsers might round off to only 10 significant digits.

The VS Solvers echo tabular data with a precision that can be set at runtime. The default precision uses 10 significant digits. Prior to May 2010, VS Solvers wrote tabular data with only 6 significant digits (the default format for programs written in the C language). Be careful about copying tabular data from Echo files, especially if they were generated with older versions. If global coordinates are used for road data, it is possible that the values were rounded off in printing, with a loss of information.

Road Friction Screens

Two libraries are available for specifying VS Road friction as a function of path S and L coordinates.

1. **Road: Friction Map, S-L Grid**. Use this screen to specify the tire/road friction as a function of S-L coordinates for roads when a rectangular grid of S-L coordinates is sufficient.
2. **Road: Friction Map, Variable Width**. Use this screen to specify the tire/road friction as a function of S-L coordinates for roads whose widths vary or have frictional changes that are not necessarily parallel or perpendicular to the reference line of the road.

The friction maps use Configurable Function controls described in the document *VS Browser (GUI and Database)* (**Help** > **Reference Manuals**).

The tire models typically support two coefficients of friction: one for lateral forces and aligning moment, and one for longitudinal force. Advanced users can import separate coefficients or define them with VS Commands. However, for routine use, the coefficient of friction is independent of direction, and can be specified as a function of S and L.

Physically, coefficients of friction between a tire and the ground always have positive values. VS Math Models follow a convention that zero and negative values can also be specified, with these interpretations:

- If friction is set to a positive value, then that value (`MU_ROAD`) is used for both the lateral and longitudinal friction level in the internal tire model.
- If friction is set to 0.0, then the tire models produce zero shear force and moment. This supports some special applications such as kinematic and compliance (K&C) test rigs where tire forces are disabled with bearing plates.

- If friction is given a negative value, then the VS Math Model will use the tire data directly (with whatever friction level applied during testing), rather than scaling the shear forces and moments. Internally `MU_ROAD` is set to `MU_REF_X` (the tire reference coefficient) for the longitudinal direction and `MU_REF_Y` for the lateral direction.

Road Rolling Resistance Screens

Two libraries are available for specifying VS Road rolling resistance coefficient as a function of path S and L coordinates.

1. **Road: Rolling Resistance Map, S-L Grid**. Use this screen to specify the road rolling resistance coefficient as a function of S-L coordinates for roads when a rectangular grid of S-L coordinates is sufficient.
2. **Road: Rolling Resistance Map, Variable Width**. Use this screen to specify the road rolling resistance coefficient as a function of S-L coordinates for roads whose widths vary or have frictional changes that are not necessarily parallel or perpendicular to the reference line of the road.

The rolling resistance maps use Configurable Function controls described in the document *VS Browser (GUI and Database)* (**Help** > **Reference Manuals**).

Datasets from either of these libraries may be linked to a Road 3D Surface dataset using one of the miscellaneous blue links on the screen. If doing so, leave the constant coefficient field blank.

The Paths/Road: Segment Builder (Legacy) Screen

This legacy screen was used to create a Reference Path in segments using arcs and straight lines. Horizontal (X-Y) data is always created; simple elevation, banking, and frictional coefficient data can be added (Figure 35) based on the drop-down list selection in the upper left corner of the screen (Figure 34).

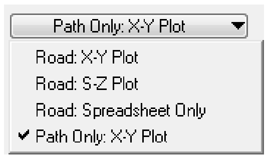


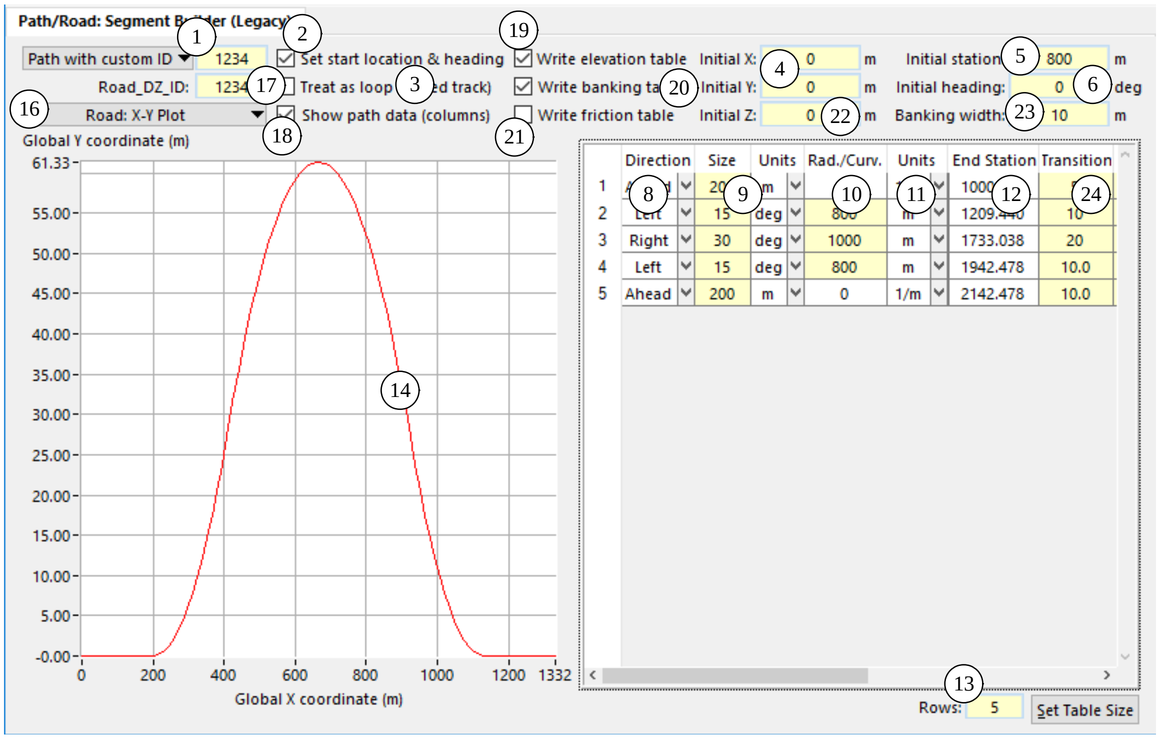
Figure 34. Options for showing plots and limiting the scope of the dataset.

Controls on this screen for building a Reference Path were described earlier in the **Reference Path** section (page 15). (These are the controls numbered ① - ⑯.) Additional controls that affect road surface elevation and friction are described below.

Specify a Reference Path Plus Road Surface Data

The drop-down list ⑯ has four options: three for a road, and one for a path only (Figure 34).

When the drop-down list ⑯ is set to anything other than **Path Only: X-Y Plot**, more options are shown on the screen that can be used to specify elevation and friction properties for a road.


Path/Road: Segment Builder (Legacy)

Path with custom ID: 1234

Road_DZ_ID: 1234

Road: X-Y Plot

[✓] Set start location & heading

[ ] Treat as loop (closed track)

[✓] Show path data (columns)

[✓] Write elevation table

[✓] Write banking table

[ ] Write friction table

Initial X: 0 m

Initial Y: 0 m

Initial Z: 0 m

Initial station: 800 m

Initial heading: 0 deg

Banking width: 10 m

	Direction	Size	Units	Rad./Curv.	Units	End Station	Transition
1	Ahead	20	m		1	1000	5
2	Left	15	deg	800	m	1209.440	10
3	Right	30	deg	1000	m	1733.038	20
4	Left	15	deg	800	m	1942.478	10.0
5	Ahead	200	m	0	1/m	2142.478	10.0

Rows: 5

Figure 35. The Segment Builder screen when used to generate road elevation or friction data.

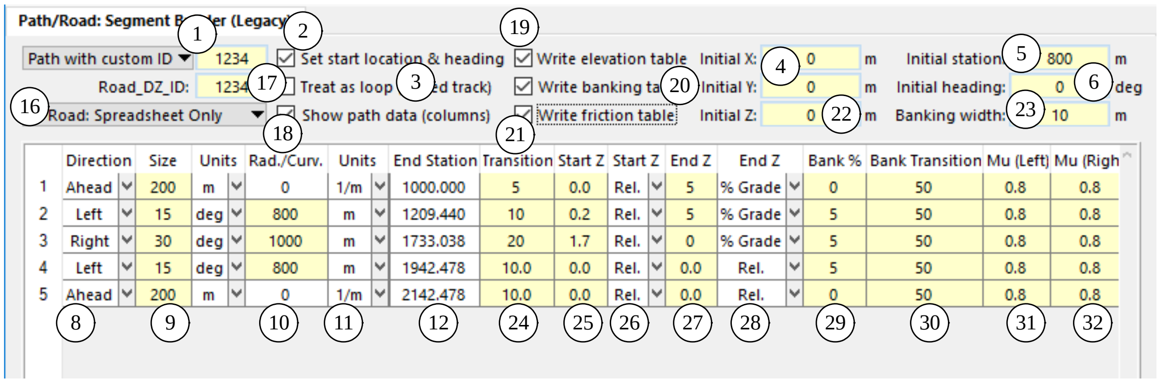
(17) Yellow field to specify an ID number for the off-center elevation layer (`Road_DZ`) (e.g., 1001). This field is shown when (1) is set to **Path with custom ID** and the box to **Write banking table** (20) is checked.

The ID number can be a positive integer that is 999 or greater, as described for other custom IDs earlier (page 7).

(18) Checkbox to hide/show columns in the table with path data. This affects the data display in the spreadsheet but does not affect how the VS Solvers will read the underlying dataset. When the box is not checked, the first five columns in the spreadsheet are hidden but still in effect. This is done to quickly view other columns that are otherwise out of view.

(19) Checkbox to write elevation table. Checking this box adds columns to the screen for specifying the Z coordinates of each segment of the road (Start Z and End Z, Figure 36). Unchecking the box means the VS Solvers are not sent any elevation data for the road from this screen, and the columns pertaining to the elevation data are hidden from view. In such a case, road elevation data can be specified using one of the other road elevation libraries, linked from the **Road: 3D Surface (All Properties)** screen.

(20) Checkbox to write banking table. Checking this box adds columns to the screen for specifying the banking properties of each segment of the road (Bank % and Bank Transition, Figure 36). In addition, the fields to specify Banking Width (23) and potentially `Road_DZ_ID` (17) are shown. When this box is unchecked, the VS Solvers are not sent any banking data for the road from this screen, and the columns pertaining to the road banking are hidden from view.



Path/Road: Segment B ler (Legacy)

Path with custom ID 1234

Road_DZ_ID: 1234

Road: Spreadsheet Only

☑ Set start location & heading

☐ Treat as loop (ed track)

☑ Show path data (columns)

☑ Write elevation table

☑ Write banking ta

☑ Write friction table

Initial X: 0 m

Initial Y: 0 m

Initial Z: 0 m

Initial station 800 m

Initial heading: 0 deg

Banking width: 10 m

	Direction	Size	Units	Rad./Curv.	Units	End Station	Transition	Start Z	Start Z	End Z	End Z	Bank %	Bank Transition	Mu (Left)	Mu (Righ
1	Ahead	200	m	0	1/m	1000.000	5	0.0	Rel.	5	% Grade	0	50	0.8	0.8
2	Left	15	deg	800	m	1209.440	10	0.2	Rel.	5	% Grade	5	50	0.8	0.8
3	Right	30	deg	1000	m	1733.038	20	1.7	Rel.	0	% Grade	5	50	0.8	0.8
4	Left	15	deg	800	m	1942.478	10.0	0.0	Rel.	0.0	Rel.	5	50	0.8	0.8
5	Ahead	200	m	0	1/m	2142.478	10.0	0.0	Rel.	0.0	Rel.	0	50	0.8	0.8

Figure 36: Path/Road: Segment Builder (Legacy) screen with extended editor

Note This screen can be used to specify simple banking of the road as it goes around turns. Other forms of banking can be set directly using other libraries.

(21) Checkbox to write friction table. Checking this box adds two columns to the screen for specifying two friction properties per segment — Mu (Left) and Mu (Right) — Figure 36. When this box is unchecked, the VS Solvers are not sent any frictional coefficient data for the road from this screen, and the columns pertaining to the road friction are hidden from view.

(22) Global Z coordinate for the road at the start of the first path segment. This field is not shown unless the checkbox for elevation data is checked.

(23) Banking width. If the road is banked, this field specifies the width of the part of the road that should be banked. Beyond this width, the terrain is flat. Values of 0.0 or below will cause the banking to be extrapolated indefinitely.

(24) Elevation transition distance. This value specifies the transitional distance for the elevation at the end of the current segment and the beginning of the next segment. More detail is in the subsection Elevation Data that follows.

Extended Editor View

As noted previously, the **Path/Road: Segment Builder (Legacy)** screen has four options for displaying the spreadsheet data: three when the intention is to represent a road, and one when the intention is to represent a path only.

Checking all of the boxes on the screen adds the columns to the spreadsheet as described above, and a horizontal scroll bar is available for viewing the added columns that are not normally in view.

As an alternative to scrolling, we can view all columns at once by selecting the option **Road: Spreadsheet Only** from the drop-down control (16) (Figure 36). In this case, the plot of the road is hidden and the spreadsheet view is extended.

Elevation Data

The five columns after End Station — Transition, Start Z, and End Z ((24) - (28)) — specify how elevation changes in each of the segments. These columns are visible only when the **Write**

elevation table box ⑲ is checked. If the box is not checked, then no information is provided to the VS Math Model from this screen about centerline elevation changes in the road.

㉔ Elevation transition distance. This value specifies the transitional distance for the elevation at the end of the current segment and the beginning of the next segment. A spline curve will be automatically drawn in this region to connect the two segments. A conceptual drawing of this is shown in Figure 37 in which this field would be the value T1.

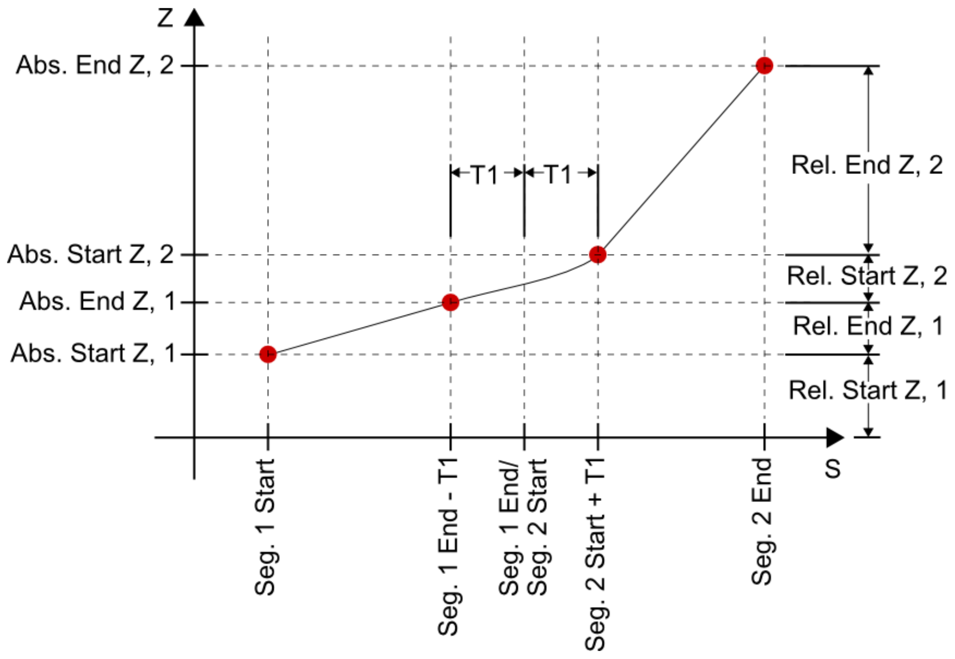


Figure 37. Conceptual drawing of the elevation description of the Segment Builder screen.

If the road loops, the transition distance for the final segment of the road applies to the first and last segments. However, if the road does not loop, the last segment has no ending transitional distance, and any value specified for the last segment is ignored.

㉕ Start Z value. This field describes the beginning Z coordinate of the segment.

The value of this field specifies the elevation after the transitional period. If the road does not loop, however, the first segment will begin at this elevation without any transitional period.

The value of this field can specify the elevation using either an absolute Z coordinate or a relative Z coordinate, depending on the selection of column ㉖. A relative Z coordinate adds the value of this field to the elevation at the end of the previous segment (before the transitional period). For the first segment, both absolute and relative starting Z coordinates behave the same.

㉖ Convention used to specify start Z value ㉕; absolute or relative.

㉗ End Z value. This field describes the Z coordinate at the end of the segment. Usually this value occurs at a station distance equal to the final station value of the segment minus the transition distance ㉔ of the segment. If the road does not loop, the value for the last row specifies the elevation of the last point in the road. Depending on the selection of column ㉘, this value may specify: (1) an absolute Z; (2) a Z coordinate relative to the segment's starting Z coordinate; or (3) a percent grade of the segment.

(28) End Z value type. This field specifies the type of data in column (27) and may be an absolute coordinate, a value relative to the start of the segment, or a percent grade for the segment.

Banking Data

Two columns specify banking information about the segments. These columns are visible only when the **Write banking table** box (20) is checked. If the box is not checked, then no information is provided to the VS Math Model about banking changes in the road.

(29) Banking percentage column. This column specifies the cross-slope of the segment. If the segment is curving either left or right, positive values of banking will make the road bank into the turn. If the segment is straight, positive values of banking will cause the road to slope down to the left.

(30) Banking transition distance. This column specifies the distance at the end of one segment and the beginning of the next segment that will be used for linearly transitioning one banking amount to another. Since the transition distance is applied to both the end of one segment and the beginning of the next, the actual distance in which the banking is transitioning is twice the value of this column. If the road loops, the final transition distance applies to the first and last segments. However, if the road does not loop, the last segment has no ending transitional distance, and any value specified is ignored.

Friction Data

Two columns specify friction information about the segments. These columns are visible only when the **Write friction table** box (21) is checked. If the box is not checked, then no information from this screen is provided about friction of the road.

Coefficients of friction always have positive values. The VS Math Models allow a value of zero to be specified for special applications such as kinematic and compliance (K&C) test rigs where tire forces are disabled with bearing plates. If you enter a negative value, `MU_ROAD` will be set to `MU_REF_X` (the tire reference coefficient) for the longitudinal direction and `MU_REF_Y` for the lateral direction. By using a negative value, you force the VS Math Model to use the tire data directly without scaling for friction.

(31) Mu (Left). This column specifies the coefficient of friction to be applied to all lateral values to the left of the centerline for this segment.

(32) Mu (Right). This column specifies the coefficient of friction to be applied to all lateral values to the right of the centerline for this segment.

Path and Road Tables with X-Y Coordinates

Three screens are used to provide tables of global X and Y coordinates used in a Reference Path; these were described earlier. In addition, two screens provide tables of X-Y-Z coordinates to provide both path and elevation information for a road:

1. **Road: X-Y-Z Coordinates of Reference Line**

2. **Road: X-Y-Z Coordinates of Edges**.

As noted earlier, a Reference Path provides an S-L coordinate system used in the VS Math Model. Because S is such a critical variable when simulating a road, the S value associated with each row in a table of X and Y coordinates is computed and shown whenever the table or plot is refreshed.

Looped Paths that use X-Y Tables

VS Solvers support two methods for handling the definition of station outside the range of the specified geometry

1. An open-ended path has station values that are extended beyond the range of the table via extrapolation.
2. A looped path requires the first and last points to be equal. Station values outside the range of the table are increased or decreased by multiples of the lap length until they are within the range of the table.

Effects of Spline Interpolation and Extrapolation

If the path is not looped, then it is a good idea to check the heading angle of the path when extrapolated outside the range of the table. If the path is used to define a road, then the best way to view the road is with the video visualization. To view the road:

1. Go to the **Road: 3D Surface (All Properties)** screen that makes use of the geometric data (horizontal and elevation), click the **Update Road Surface 3D Shape Files** button ⑨ (Figure 27, page 33), and then click the **Video Preview** button ⑬.
2. If the road does not extend far enough in both directions, then follow link ⑨ (Figure 27) to go to the linked **Road: Animator Surface Shapes** screen. Change the start and stop station numbers, then go back to the **Road: 3D Surface (All Properties)** screen and repeat step 1.

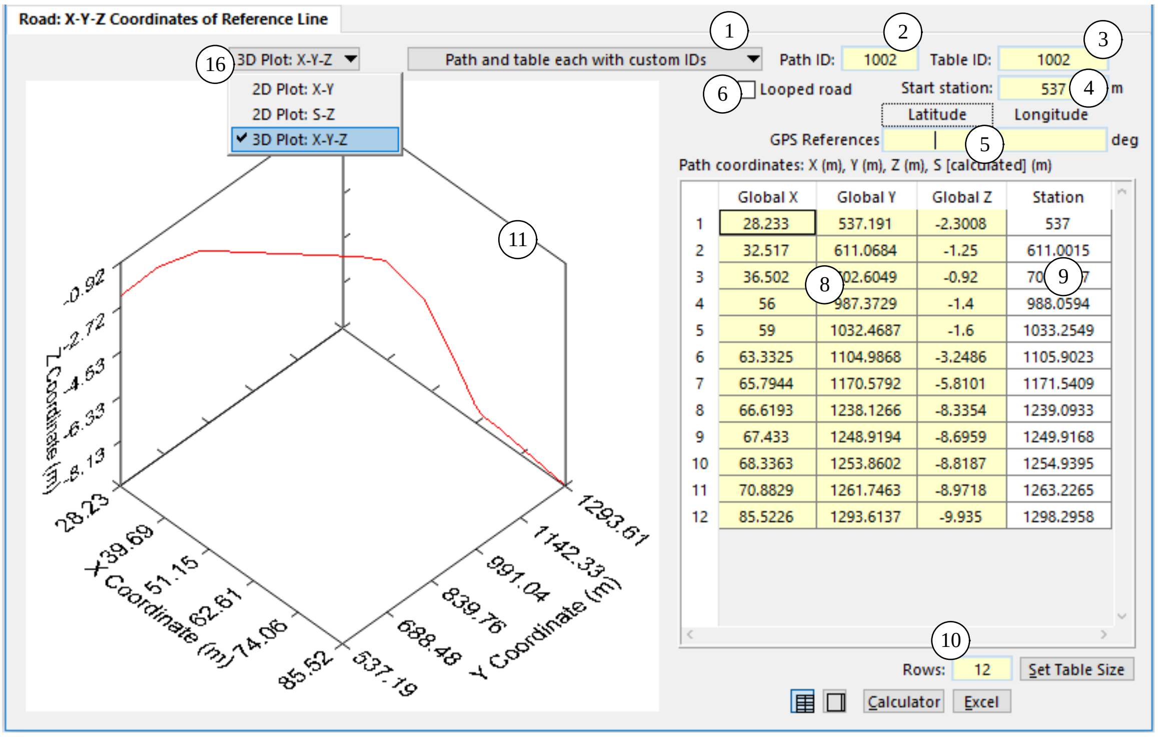
Road: X-Y-Z Coordinates of Reference Line

Use this screen (Figure 38) to define both the path used for a road and the associated elevation with a table of X-Y-Z values.

Most of the controls on this screen were described in an earlier subsection (**Path and Road Tables with X-Y Coordinates,** page 17), with the addition of another column in the table to provide Z coordinates for each point ⑧, and a control to specify a type of plot ⑯.

Options to import GPS coordinates from Atlas are not available on this screen because the CSV file format does not include elevation. To import GPS data that includes altitude, consider using the **Scene: External Import** screen.

⑯ Drop-down list to specify the information plotted on the screen. Options include plotting the horizontal (X-Y) data, plotting the elevation (S-Z) data, or plotting a 3D view of the centerline as shown in Figure 38 in the graphic area. If the 3D view is selected, you can pan using the left-mouse button, and zoom using Ctrl plus the left mouse button.



Path coordinates: X (m), Y (m), Z (m), S [calculated] (m)

	Global X	Global Y	Global Z	Station
1	28.233	537.191	-2.3008	537
2	32.517	611.0684	-1.25	611.0015
3	36.502	702.6049	-0.92	70[illegible]7
4	56	987.3729	-1.4	988.0594
5	59	1032.4687	-1.6	1033.2549
6	63.3325	1104.9868	-3.2486	1105.9023
7	65.7944	1170.5792	-5.8101	1171.5409
8	66.6193	1238.1266	-8.3354	1239.0933
9	67.433	1248.9194	-8.6959	1249.9168
10	68.3363	1253.8602	-8.8187	1254.9395
11	70.8829	1261.7463	-8.9718	1263.2265
12	85.5226	1293.6137	-9.935	1298.2958

Figure 38. The Road: X-Y-Z Coordinates of Reference Line screen.

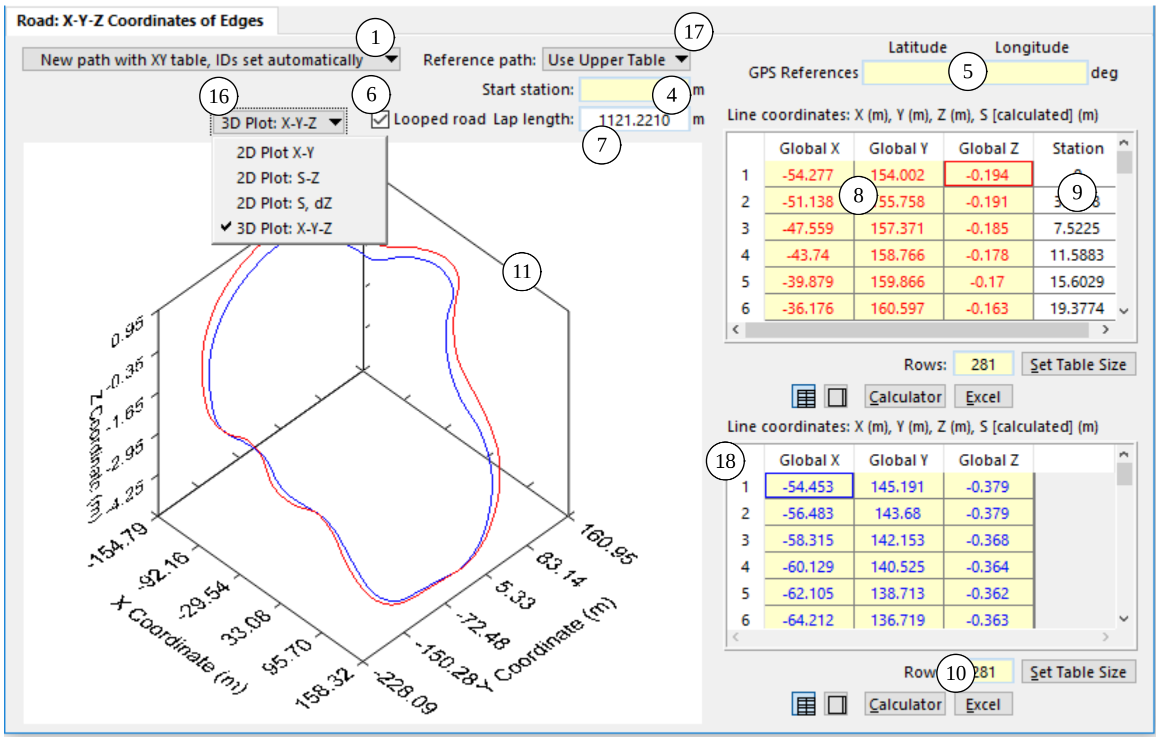
Warning When all three coordinates are provided for each point, the accuracy of the Z coordinates can be a problem. It is usually acceptable if the X and Y coordinates are accurate to a few centimeters. However, for typical paved roads, the Z coordinates should be correct to within a fraction of a millimeter. If data are obtained from GPS sources, the Z coordinates should be smoothed before being put into the table. Otherwise, the normal measurement noise appears to a vehicle math model as extremely high roughness.

Road: X-Y-Z Coordinates of Edges

This screen (Figure 39) can be used to define a Reference Path, vertical geometry, and cross-slope (banking) of a road from global X-Y-Z coordinates of two paths on the road. Such data may be obtained from GPS measurements of the two edges of the road, for example.

Warning As noted above, accuracy with Z coordinates can be a problem with GPS data. Because of this problem, this screen is not often used.

You select one of the two edges to be used as the Reference Path of the road (17, 18). To get the cross-slope of the road, the screen automatically fits a variable-width table to the data. Although the edge chosen to be the Reference Path is reproduced precisely, the process of fitting the variable-width table to the data involves linearly interpolating between data points of the other edge.



New path with XY table, IDs set automatically

Reference path: Use Upper Table

Start station: m

☑ Looped road Lap length: 1121.2210 m

GPS References Latitude Longitude deg

Line coordinates: X (m), Y (m), Z (m), S [calculated] (m)

	Global X	Global Y	Global Z	Station
1	-54.277	154.002	-0.194	0
2	-51.138	155.758	-0.191	[illegible]
3	-47.559	157.371	-0.185	7.5225
4	-43.74	158.766	-0.178	11.5883
5	-39.879	159.866	-0.17	15.6029
6	-36.176	160.597	-0.163	19.3774

Rows: 281

Line coordinates: X (m), Y (m), Z (m), S [calculated] (m)

	Global X	Global Y	Global Z
1	-54.453	145.191	-0.379
2	-56.483	143.68	-0.379
3	-58.315	142.153	-0.368
4	-60.129	140.525	-0.364
5	-62.105	138.713	-0.362
6	-64.212	136.719	-0.363

Rows: 281

Figure 39. The Road: X-Y-Z Coordinates of Edges screen.

This screen has all the controls of the screen described in the previous subsection (**Road: X-Y-Z Coordinates of Reference Line,** page 49), with the following additions:

(17) Drop-down list to select which table (upper or lower) will be used as the Reference Path of the road.

(18) Lower table of X-Y-Z coordinates representing the second edge of the road. Unlike the other Path screens in which we define the centerline (i.e., the road Reference Path), and any LTARG datasets for the Driver Model and moving objects would represent a lateral offset with respect to that centerline, this screen defines the two edges of a road, from which we choose one edge or the other to become the Reference Path ((19)). An LTARG dataset would then represent a lateral offset to either of these edges.

1. If the road is set to be looped ((5)), the table represents the inner edge of the road as is shown in the figure.
2. If the road is not looped, such as a straight road with cross-slope, and the direction of travel is increasing station, this lower table represents the right edge of the road.

Connecting VS Road Surfaces

When multiple VS Roads are defined for a simulation, there are several ways to determine which surface is used for contact between the vehicle tires and the ground, as well as elevation and 3D orientation of moving objects. The road surface that is initially used by the vehicle's tires is specified with the parameter CURRENT_ROAD_ID, and each moving object can be associated with a road using the parameter ROAD_ID_OBJ(*i*), where *i* is the object number.

Points of Interest for Boundaries

If road surfaces are connected, the switching from one surface to the other is handled automatically by specifying the boundaries of the road. The VS Solver internally keeps track of the road surface for *points of interest*, by updating several variables related to the contact of the point with the surface. These internal variables include the Road ID and the station S in the road Reference Path. There are three groups of points that are automatically monitored:

1. A point is tracked for each sprung mass in the vehicle model (at the aerodynamic reference point). The parameter `CURRENT_ROAD_ID` is updated when the point for the lead unit changes surfaces, and the output variable `Sta_Road` is set to the station on the Reference Path for the road specified with the `CURRENT_ROAD_ID` parameter.
2. A tire/ground contact point is tracked for each tire. For each point, the output variable `RdID_`*tire* provides the road ID (e.g., `RdID_L2` is the current road ID for tire L2), and the output variable `S_Rd_`*tire* provides the road station coordinate (e.g., `S_Rd_L2`).
3. A point is tracked for each user-defined moving object (created with datasets from the **Multiple Moving Objects** and **Single Moving Object (Custom)** libraries). The parameter `ROAD_ID_OBJ`(*o*) for object *o* is updated to the current road ID used by the object when the object crosses a boundary, and the output variable `S_RdO_`*o* provides the station coordinate (e.g., `S_RdO_5`).

When a point of interest crosses a boundary, the VS Solver automatically changes the associated road surface and road station associated with that point.

For example, Figure 40 shows video and plots for a CarSim vehicle entering a roundabout intersection that involves five adjacent roads, with ID numbers 1000 – 1004. The upper-left plot indicates that the vehicle's tires are currently on road surface 1001 but will move to road 1000 (at about t = 6 s) and then to 1003 (at about t = 17 s). The upper-right plot shows the station for each tire (currently around S = 60 m), and indicates adjustments are made as the tires change surfaces at about t = 7s and again at about t = 18 s.

The driver preview points, shown as blue spheres in the video, are moving objects that are also tracked automatically. The lower plots show that point 10 is currently on road 1000 while point 5 is still on road 1001. Both will be on road 1000 from about 6 s to 17 s (this is the circle of the roundabout) and will finish on road 1003.

Boundary Definitions

Each road surface has four potential boundaries: start, end, left, and right. These boundaries are used to check the location of all current points of interest. Each boundary is defined with one of three options:

1. There is no boundary. In this case, the road surface extends infinitely in the given direction (start, end, left, right), and no other information is needed.
2. There is a boundary, and the simulation will stop automatically if any point of interest crosses the boundary.
3. There is a boundary, and, if a point of interest crosses the boundary, then the road surface for the point of interest is switched automatically.

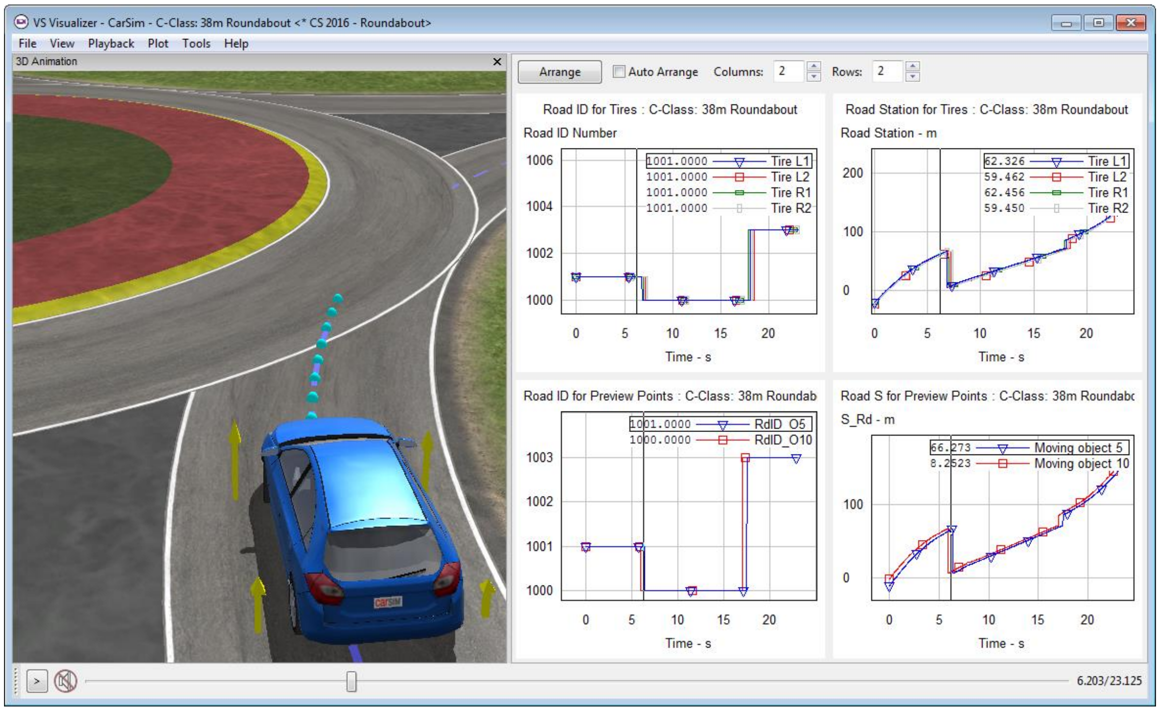

VS Visualizer - CarSim - C-Class: 38m Roundabout <* CS 2016 - Roundabout>
File View Playback Plot Tools Help
3D Animation
Arrange
Auto Arrange
Columns: 2
Rows: 2
Road ID for Tires : C-Class: 38m Roundabout
Road ID Number
Tire L1
Tire L2
Tire R1
Tire R2
Time - s
Road Station for Tires : C-Class: 38m Roundabout
Road Station - m
Road ID for Preview Points : C-Class: 38m Roundabout
RdID_O5
RdID_O10
Road S for Preview Points : C-Class: 38m Roundabout
S_Rd - m
Moving object 5
Moving object 10
6.203/23.125

Figure 40. Automatic adjustment of Road ID and station for tires and preview points.

If there is a boundary, the start and end boundaries are defined with minimum and maximum station values, respectively. The left and right boundaries are defined with maximum and minimum L values, respectively.

If there is a boundary that involves switching to a different road, the new road may have a separate S-L coordinate system. When entering the new road, it is necessary that the VS Solver be able to estimate the station in the new S-L coordinate system.

The start and end boundaries are each defined with three parameters involving station S, and the left and right boundaries are each defined with three Configurable Function datasets associated with both S and L. Table 2 shows the names of the parameters (P in the last column) and functions (F in the last column).

Table 2. Parameters and Configurable Functions for road boundaries.

Boundary	**New ID**	**Limit**	**New S**	**P/F**
End	`ROAD_SMAX_NEW_ID`	`ROAD_SMAX`	`ROAD_SMAX_NEW_S`	P
Start	`ROAD_SMIN_NEW_ID`	`ROAD_SMIN`	`ROAD_SMIN_NEW_S`	P
Left/Right	`ROAD_NEW_ID`	`ROAD_L_BOUNDARY`	`ROAD_NEW_S`	F

The three configurable functions in the third row of the table use keywords with two indices: one for the road (`IROAD`) and one for the side (`ISIDE`: 1 = left side, 2 = right side). Libraries with datasets for these functions are:

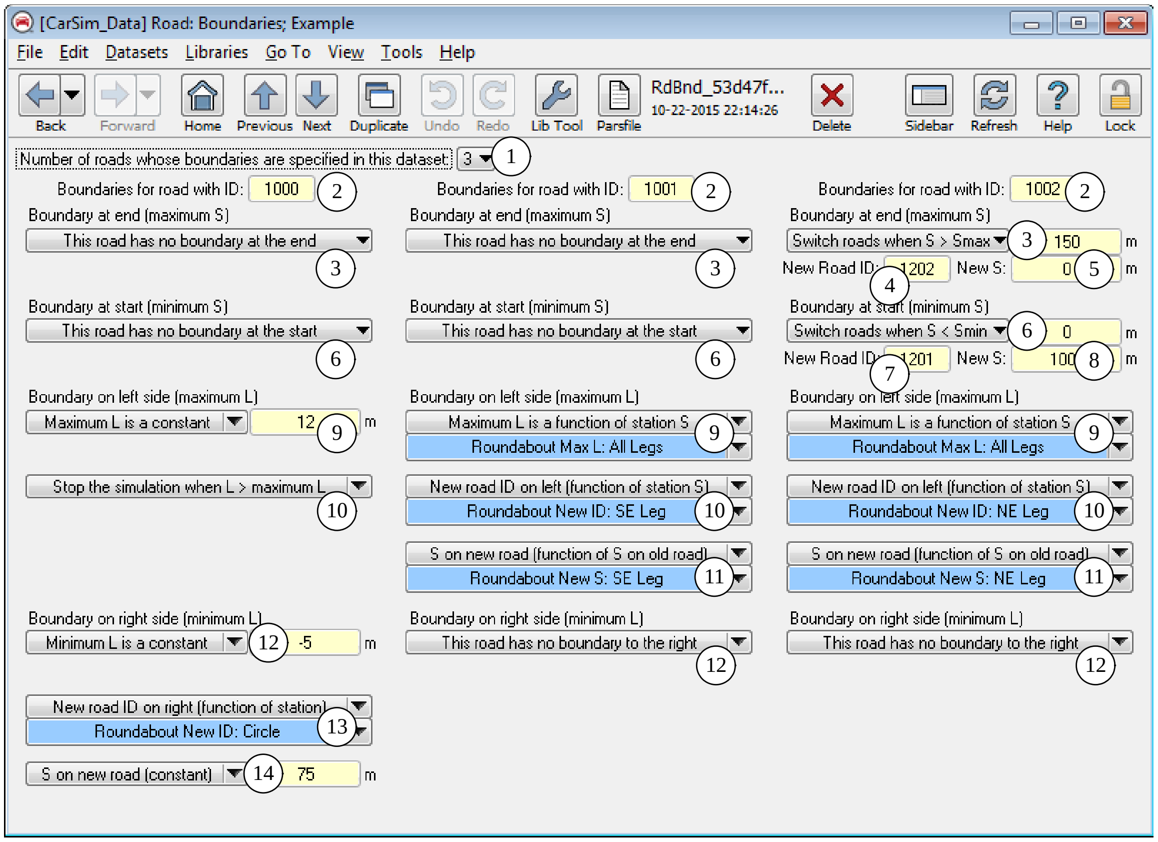
1. **Road Boundary: Lateral Edge**
2. **Road Boundary: New Road ID**

3. **Road Boundary: New Station**

When viewing an Echo file, note that the parameters involving start and end boundaries are listed along with other parameters for each road. The Configurable Functions for the left and right boundaries are listed later in the file, in alphabetical order with other Configurable Function tables and parameters. Each of these Configurable Functions support 400 datasets per simulation (two per road).

The Road: Boundaries Screen

The boundaries can be set using the Road: Boundaries library (Figure 41). The screen is organized into three columns of settings, where each column defines boundaries for one road surface.



[CarSim_Data] Road: Boundaries; Example
File Edit Datasets Libraries Go To View Tools Help
Back Forward Home Previous Next Duplicate Undo Redo Lib Tool Parsfile
RdBnd_53d47f...
10-22-2015 22:14:26
Delete Sidebar Refresh Help Lock

Number of roads whose boundaries are specified in this dataset: 3 (1)

Boundaries for road with ID: 1000 (2)
Boundary at end (maximum S)
This road has no boundary at the end (3)
Boundary at start (minimum S)
This road has no boundary at the start (6)
Boundary on left side (maximum L)
Maximum L is a constant 12 (9) m
Stop the simulation when L > maximum L (10)
Boundary on right side (minimum L)
Minimum L is a constant (12) -5 m
New road ID on right (function of station)
Roundabout New ID: Circle (13)
S on new road (constant) (14) 75 m

Boundaries for road with ID: 1001 (2)
Boundary at end (maximum S)
This road has no boundary at the end (3)
Boundary at start (minimum S)
This road has no boundary at the start (6)
Boundary on left side (maximum L)
Maximum L is a function of station S (9)
Roundabout Max L: All Legs
New road ID on left (function of station S)
Roundabout New ID: SE Leg (10)
S on new road (function of S on old road)
Roundabout New S: SE Leg (11)
Boundary on right side (minimum L)
This road has no boundary to the right (12)

Boundaries for road with ID: 1002 (2)
Boundary at end (maximum S)
Switch roads when S > Smax (3) 150 m
New Road ID: 1202 (4) New S: 0 (5) m
Boundary at start (minimum S)
Switch roads when S < Smin (6) 0 m
New Road ID: 1201 (7) New S: 100 (8) m
Boundary on left side (maximum L)
Maximum L is a function of station S (9)
Roundabout Max L: All Legs
New road ID on left (function of station S)
Roundabout New ID: NE Leg (10)
S on new road (function of S on old road)
Roundabout New S: NE Leg (11)
Boundary on right side (minimum L)
This road has no boundary to the right (12)

Figure 41. The Road: Boundaries library screen.

Alert The recommended linking order for **Road: Boundaries** datasets with respect to the **Road: 3D Surface (All Properties)** datasets is based on whether there is a single road or multiple roads.

If the boundaries apply only for a single road (i.e., they only involve stopping the simulation, but not transfer to another surface), then it is OK to link to a **Road: Boundaries** dataset from the **Road: 3D Surface (All Properties)** dataset.

If there are multiple roads and the **Road: Boundaries** datasets are used to switch between roads, such as what is shown in Figure 41, then it is best to group the roads together using one of the Generic screens (e.g., **Generic Data Group** or **Generic Group of Links**), then link to the **Road: Boundaries** datasets after the roads. This ensures that the IDs for the roads are defined prior to them being used on the **Road: Boundaries** datasets. For example, the VS Solver will generate an error if a road with ID 1234 is specified on this screen and no road with that ID can be found.

① Drop-down control to specify the number of road surfaces whose boundaries are defined in the dataset. This will show 1, 2, or 3 columns of settings, one per road.

② Yellow field to specify the `ROAD_ID` for an existing road. All the settings in the column (under this field) will apply to this road. The screen Parsfile uses the command `GET_IROAD`, where `IROAD = GET_IROAD(id)`, as in `IROAD = GET_IROAD(1234)`. If there is no road with the specified ID, the VS Solver will report an indicating that is cannot locate a road with that ID, and cancel the simulation.

Boundaries at the Start and End (Limits in Station)

The start and end boundaries for a road are defined in terms of minimum (start) and maximum (end) station S.

③ Drop-down control to specify an option for a boundary at the end of the road (Figure 42). If a boundary exists, it is located at a specified maximum station S.

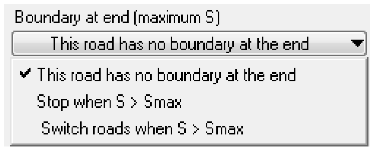

Boundary at end (maximum S)
This road has no boundary at the end
✓ This road has no boundary at the end
Stop when S > Smax
Switch roads when S > Smax

Figure 42. Drop-down control to choose an option for a boundary at the end of a road.

The three options are:

1. There is no boundary at the end of the road (there is no maximum limit to station S on this road). When this option is selected, the parameter `ROAD_SMAX_NEW_ID` is set to zero.
2. There is a boundary, and when any moving point of interest reaches the boundary, the simulation stops. When this option is selected, the parameter `ROAD_SMAX_NEW_ID` is set to -1. An adjacent yellow field is shown (see the third column in Figure 41) to specify the maximum station (keyword = `ROAD_SMAX`).
3. There is a boundary, and when any moving point of interest reaches the boundary, the math model automatically switches the road for the point to another one. An adjacent yellow field is shown (see the third column in Figure 41) to specify the maximum station (keyword = `ROAD_SMAX`). Two other fields are shown: one to specify the ID of the new road ④, and the other to specify an estimate for the station on the new road ⑤.

④ Yellow field to specify the `ROAD_ID` for an existing road (keyword = `ROAD_SMAX_NEW_ID`). If the station of a point of interest exceeds the specified maximum station ③, the VS Solver will switch to the road with this ID. If there is no road with the specified ID, the VS Solver will report an error and cancel the simulation.

⑤ Yellow field to specify the initial station for a point of interest if it changes from the current road ② to a new road ④ when exceeding the maximum station ③. The keyword for the new station is `ROAD_SMAX_NEW_S`. This only needs to be an approximation; once the point of interest is located on the new road, the estimate provided here is replaced with the actual value of station S on the new road.

⑥ Drop-down control to specify an option for a boundary at the start of the road (Figure 43). If a boundary exists, it is located at a specified minimum station S.

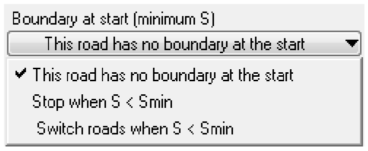

Boundary at start (minimum S)
This road has no boundary at the start
✓ This road has no boundary at the start
Stop when S < Smin
Switch roads when S < Smin

Figure 43. Drop-down control to choose an option for a boundary at the start of a road.

The options here are similar to those for the end of a road ③, except that instead of checking for station of a point of interest exceeding a specified maximum limit, the checking is done to see if the station is less than a specified minimum limit (keyword = `ROAD_SMIN`).

The three options are:

1. There is no boundary at the start of the road. When this option is selected, the parameter `ROAD_SMIN_NEW_ID` is set to zero.
2. There is a boundary, and when any moving point of interest reaches the boundary, the simulation stops. When this option is selected, the parameter `ROAD_SMIN_NEW_ID` is set to -1. An adjacent yellow field is shown (see the third column in Figure 41) to specify the minimum station (keyword = `ROAD_SMIN`).
3. There is a boundary, and when any moving point of interest reaches the boundary, the math model automatically switches the road for that point to another one. An adjacent yellow field is shown (see the third column in Figure 41) to specify the minimum station (keyword = `ROAD_SMIN`). Two other fields are shown: one to specify the ID of the new road ⑦, and one to specify an estimate for the station for the point on the new road ⑧.

⑦ Yellow field to specify the `ROAD_ID` for an existing road (keyword = `ROAD_SMIN_NEW_ID`). If the station of a point of interest exceeds the specified minimum station ⑥, the VS Solver will switch to the road with this ID. If there is no road with the specified ID, the VS Solver will report an error and cancel the simulation.

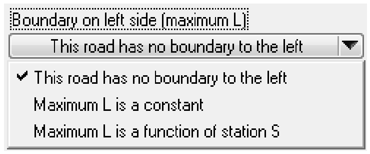
⑧ Yellow field to specify the initial station for a point of interest if it changes from the current road ② to a new road ⑦ when crossing the boundary at the minimum station ⑥. The keyword for the new station is `ROAD_SMIN_NEW_S`.

Boundaries on the Left Side (Maximum L)

The left and right boundaries for a road are defined in terms of maximum and minimum lateral position L. Given that roads are typically much longer than they are wide, and that dimensions may change over the length with S, the left and right boundary properties are defined as Configurable Functions of station S.

The three functions, identified in Table 2 (page 53), are automatically indexed to the road and side using system parameters `IROAD` and `ISIDE`, with `ISIDE` = 1 (left side) for these settings.

(9) Drop-down control to specify an option for a boundary for the left side of the road (Figure 44). If a boundary exists, it is defined by a maximum value of L, which can be a constant or a function of station S.


Boundary on left side (maximum L)

This road has no boundary to the left

✓ This road has no boundary to the left

Maximum L is a constant

Maximum L is a function of station S

Figure 44. Drop-down control to choose an option for locating an edge boundary.

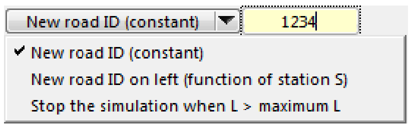
There are three options:

1. There is no boundary on the left side of the road (there is no maximum limit to L on this road). When this option is selected, the Configurable Function for the road ID is set to a constant `ROAD_NEW_ID_CONSTANT` with a value of zero.

2. There is a boundary, and it is defined with a constant L value. In this case, another control is shown to specify what happens when the limit in L is reached (10), and an adjacent yellow field is shown (see the first column in Figure 41) to specify the maximum L (keyword = `ROAD_L_BOUNDARY_CONSTANT`).

3. There is a boundary, and it is defined with a Configurable Function that calculates the maximum L as a function of S. In this case, another control is shown to specify what happens when the limit in L is reached (10), and a link under the control is made to a dataset from the **Road Boundary: Lateral Edge** library (see columns 2 and 3 in Figure 41) to calculate the maximum L (keyword = `ROAD_L_BOUNDARY`).

(10) Drop-down control to specify what happens when a point of interest reaches the left boundary of the road (Figure 45).


New road ID (constant)

1234

✓ New road ID (constant)

New road ID on left (function of station S)

Stop the simulation when L > maximum L

Figure 45. Drop-down control to choose an option for reaching an edge boundary.

There are three options:

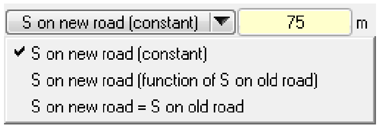
1. When any moving point of interest reaches the specified value of L, the math model automatically switches the road for that point to another road whose ID is specified in

an adjacent yellow field (Figure 45, keyword = `ROAD_NEW_ID_CONSTANT`). Another control is shown to specify how station is estimated on the new road (11).

2. When any moving point of interest reaches the boundary, the math model automatically switches the road for that point to another one whose ID is calculated as a function of station using a linked dataset from the library **Road Boundary: New Road ID**. The link for the dataset is just under the control, as shown for columns 2 and 3 in Figure 41. Another control is shown to specify how station is estimated on the new road (11).

3. When any moving point of interest reaches the boundary, the math model automatically stops. When this option is selected, `ROAD_NEW_ID_CONSTANT` is set to –1.

(11) Drop-down control to specify station for a point of interest is set if it crosses the left boundary and goes to a new road (Figure 46). This control is only shown when a left boundary has been defined, and crossing the boundary goes to a new road.


S on new road (constant) 75 m

✓ S on new road (constant)

S on new road (function of S on old road)

S on new road = S on old road

Figure 46. Drop-down control to choose an option for setting S on a new road.

There are three options, all involving the configurable function `ROAD_NEW_S`:

1. When a point of interest reaches the boundary and goes to a new road, the math model automatically sets the station for the point on the new road using the value specified in an adjacent yellow field (Figure 46, keyword = `ROAD_NEW_S_CONSTANT`).

2. When a point of interest reaches the boundary and goes to a new road, the math model automatically calculates the station using the function `ROAD_NEW_S`. A link for the dataset from the **Road Boundary: New Station** library is just under the control, as shown for columns 2 and 3 in Figure 41.

3. When a point of interest reaches the boundary and goes to a new road, that station on the new road is the same as the station on the old road. In the math model, the new station is calculated by setting `ROAD_NEW_S_COEFFICIENT` to one.

Boundaries at the Right (Minimum L)

The controls used to define the right edge boundary are conceptually the same as those used for the left, with these differences:

1. Instead of considering the case where the L coordinate for a point of interest is greater than a maximum limit, the boundary is defined when L is less than a minimum limit.

2. When using the Configurable Functions `ROAD_NEW_ID`, `ROAD_L_BOUNDARY`, and `ROAD_NEW_S`, the system parameter `ISIDE` is set to 2 (right) rather than 1 (left).

(12) Drop-down control to specify an option for a boundary for the right side of the road. This has the same options as offered for the left side (9) (Figure 41), except the limits are defined with minimum L values rather than maximum.

(13) Drop-down control to specify what happens when a point of interest reaches the right boundary of the road. This has the same options as offered for the left side (10) (Figure 41) except the new ID is for the road on the right side.

(14) Drop-down control to specify station for a point of interest is set if it crosses the right boundary and goes to a new road. This has the same options as offered for the left side (11) (Figure 41, page 54).

Guidelines for Setting VS Road Boundaries

The ID numbers for the adjacent roads at the start, end, left, and right all have default values of zero, indicating that the road has no boundaries. When working with road descriptions from databases in older versions (prior to 2016), these default settings provide the same interpretation as existed before boundaries were added to BikeSim, CarSim, and TruckSim.

Lateral Boundaries that Stop the Simulation

If an ID number for an adjacent road is set to –1 (or any other negative number), the simulation will stop when any point being monitored (tires, moving objects) crosses the boundary. This can be useful for roads with significant curvature, where it is possible that the steering controller for the simulated vehicle will exceed the limit conditions for the vehicle. For example, CarSim and TruckSim include example steady-state circle tests defined by ISO that characterize the steady-state cornering behavior up to the limit conditions. Some of the simulations involve travelling on a circular road at increasing speed, using the Closed-Loop Path-Following steering controller to follow the road reference line. For these tests, the lateral limits for the road are set such that when the vehicle reaches the limit, a tire will cross the lateral boundary and the simulation will automatically stop.

Even for tests where the conditions are not tightly defined, a lateral boundary can stop lengthy runs when a vehicle has reached limit conditions and left the road. This is also helpful for driving simulators, where beginning drivers might lose control or try to take the vehicle into areas of a virtual proving ground where the vehicle is not intended to be.

Be aware that the simulation will stop the first time any point of interest goes out of bounds. This will typically be one of the tire contact points. In setting the boundary limits, be sure to take the width of the vehicle into account.

If the simulation includes traffic vehicles represented with moving objects in adjacent lanes, then the boundaries must be set far enough to account for those lanes.

Longitudinal Boundaries that Stop the Simulation

Some simulation procedures involving stopping the simulation where a certain station is reached. Defining a boundary at the limit station can also do this. However, given that the solvers already have options to stop at a specified station, this might not provide any advantage.

Remember that the simulation will stop when the first point of interest goes out of bounds. Some examples use moving objects to visualize preview points for the target path or for lane edge detection. In these examples, if the road has a maximum station boundary, the simulation will stop when the first preview point hits the boundary.

Boundaries Between Adjacent Road Surfaces

The main purpose of the road boundaries is to connect road surfaces together to support the automatic tracking of points on tires and moving objects, as described earlier, so they will have the correct elevation and slope information. In the case of tire points, the friction and rolling resistance coefficients also change with the road surface.

Be aware that if multiple roads are flat and have the same friction properties, then the VS Solver might not need to change surfaces. If multiple roads are defined with different paths and visual properties, but no changes in elevation or friction, then the calculations done by the VS Solver for tire contact and moving object visualization will not be affected by changes in the road surfaces. Please review the discussion earlier in the subsection *Using Road Surfaces* (page 28) for options for handling networks of flat roads.

When connecting adjacent roads, the main concern is maintaining a consistent surface description with respect to elevation changes.

1. When two adjacent roads are connected at the starts and ends, then the connection can be relatively simple, using the station-based parameters.
2. If two adjacent roads follow the same path, but with different lateral positions, the connection can again be simple; the boundary for each is specified with a constant maximum (left) or minimum (right) lateral distance.
3. When adjacent roads do not follow the same paths, then the description of the boundary connections for both may require some thought. An important consideration is providing the correct elevation information to tires when they cross the boundary. Ideally, the elevations from the two adjacent surfaces are similar at the boundary. If it is not easy to define exact functions of L vs. S for the boundaries, approximate regions can be used if the area of interest has the same elevation.

As a general guideline, boundaries for adjacent roads that do not use the same coordinate systems do not have to match if the elevations for both roads are the same. If the boundaries for the two adjacent roads are not geometrically the same, then either there will be a gap between the boundaries, or some overlap. Try to ensure that there is overlap.

For example, Figure 47 shows two sets of boundaries for two roads (1003 and 1004) that curve away from each other. In the example on the left side, the boundary for each road follows the curve, as would be obtained with a boundary defined with a constant L coordinate. In the part where the roads are straight, the boundaries match perfectly. However, as the roads curve away, a gap exists between the boundaries. If, at some time in a simulation, a point of interest that is on road 1003 crosses the boundary in the curved part, the road ID associated with the point is automatically changed to 1004. Then, in the next time step, the same point is found to be outside the boundary of road 1004, and is changed to 1003. As long as the point is located in the gap, the road ID associated with the point will be changed each time step, back and forth.

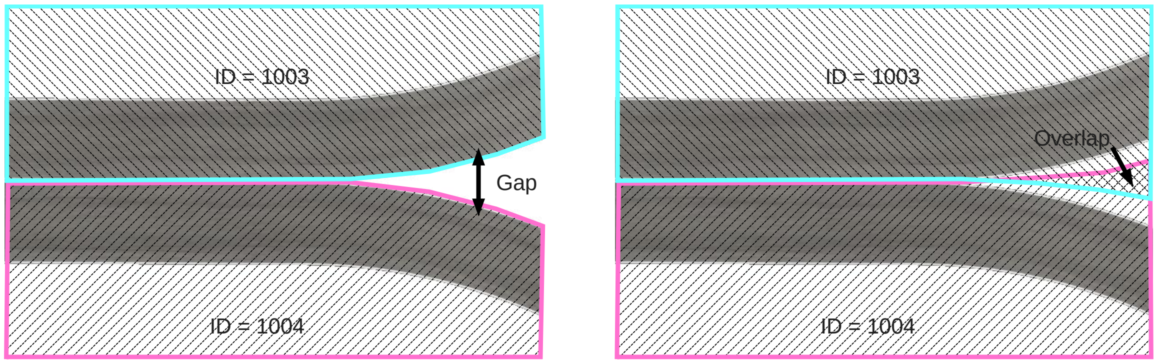


Figure 47. Example showing gap and overlap between two curved roads.

Now consider the example boundaries shown in the right-hand side of Figure 47, where they overlap in the curved section. If, at some time in a simulation, a point of interest that is on road 1003 goes into the overlap region, the road ID for the point remains set for road 1003, because it has not yet reached the boundary. When the point eventually reaches the boundary in 1003, the road ID is changed to 1004. If the point is well within the boundary for 1004, the road ID will not change back instantly.

Example: Roundabout Intersection

Roads 1003 and 1004 shown in Figure 47 are two of the five roads used to define a roundabout intersection that was shown earlier (Figure 40, page 53). This roundabout is used again as an example (Figure 48) to show how boundaries are defined for all five roads that are connected for this intersection.

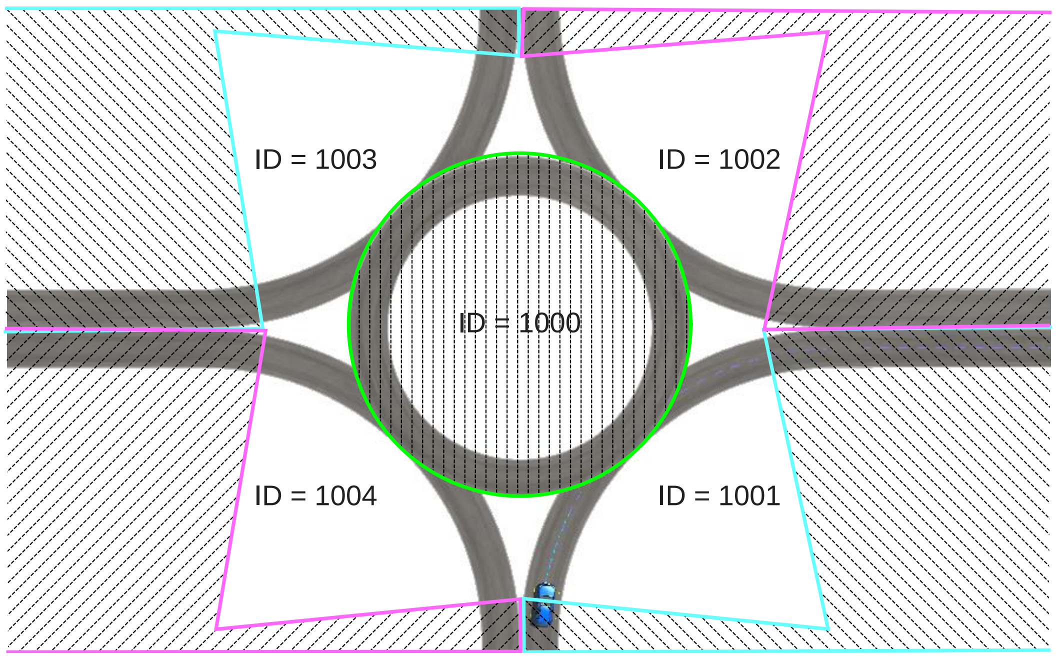


Figure 48. Boundaries for three of the five roads in a roundabout example.

Although the geometry connecting the circle to each “leg” might appear complicated, the entire region in the figure without cross-hatching has zero elevation. Therefore, the location of the boundary is not critical, as long as there is some overlap between adjacent roads.

The roundabout circle (ID = 1000) has a centerline with a 16-m radius. This circle has station going from 0 to 100.5 m, travelling counter-clockwise as is expected for driving on the right (e.g., USA). The elevation cross section includes a curb and side slope on the pavement. However, outside a limit of L < -2.1 m (an 18.1-m radius circle), the elevation is zero. In the figure, the vertical cross-hatching shows graphically the region where the elevation is not zero.

The other four road datasets (ID = 1001 - 1004) each describe a single pavement lane. All have identical geometry, other than the starting heading and X-Y coordinates. They all approach the intersection with a straight segment, then turn to the right for 90°, then exit with another straight segment. Each starts with station = 0 m, goes straight until S = 50 m, turns right until 90°, which is about S = 100 m, and then goes straight until the end at S = 150 m. The leg is completely overlapped with the roundabout circle at S = 75 m.

At the left edge (L = 2.1 m), the elevation is always 0. In the straight sections, the elevation of the right edge (at L = -2.1 m) is -0.2 m. Beyond that (L < -2.1), the elevation remains constant at -0.2 m. The elevation cross-sections transition into and out of the curve, starting 5 m before the curve and continuing 5 m after the start. In the main part of the curve, the road is flat with elevation = 0.

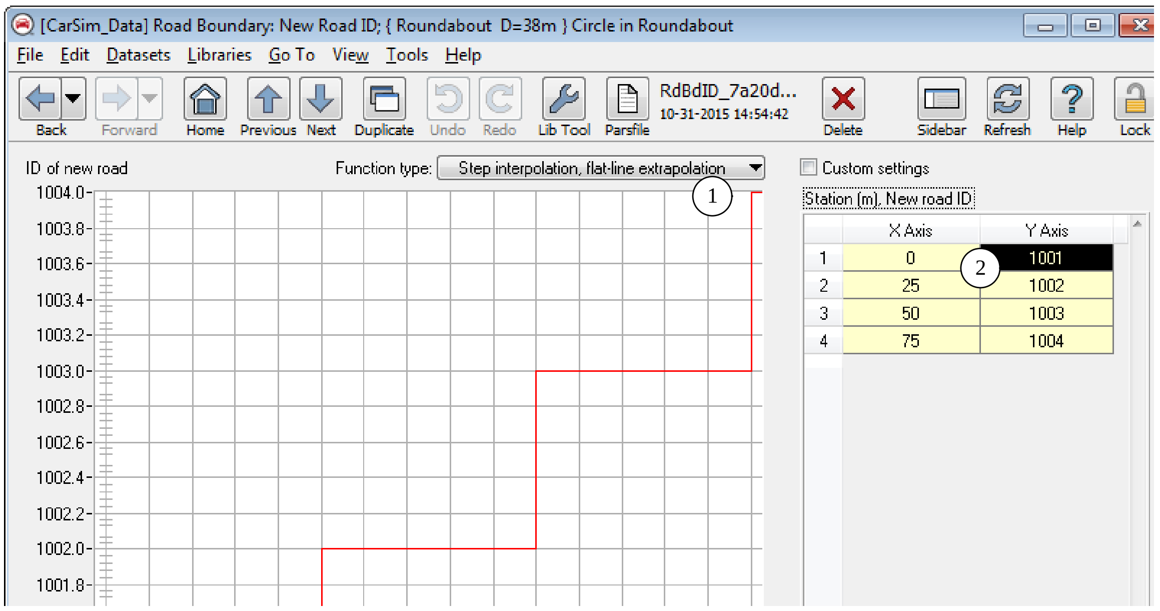
In looking at the figure, the regions with cross-hatching involve non-zero elevation. There are two kinds of boundaries with respect to elevation.

1. The roundabout circle (ID = 1000) connects with the other four road datasets.
2. Each of the legs of the intersection (ID = 1001 – 1004) has two straight sections with elevation information. The boundary is always with a straight section for another leg. For example, as shown earlier, the left part of the figure requires a boundary between roads 1003 and 1004. Following road 1004 past the circle, the bottom part requires a boundary between roads 1004 and 1001.

In the case of the circle, the boundary from the circle to the legs can be anywhere outside the hatched area in the figure. The first column of settings shown in the **Road Boundaries** screen in Figure 41 (page 54) was used for this circle. The road is looped, so there are no minimum or maximum station values. The limit for the outside (travelling counter-clockwise) is L = -5 m, well beyond the visible lane edge at -2.1 m. A linked dataset defines the road ID as a function of ring station when this boundary is crossed (Figure 49). Notice that the function is set to use Step Interpolation ①. With the four rows in the table, the only calculated values will be the integers 1001 – 1004 ②.

For each leg, the boundaries were set as shown earlier (Figure 41) in column 2 for road 1001. There is no boundary set for the right edge. For the left edge, the limit is at 2.1 m for the straight sections (0 to 50 to enter, and 100 to 150 to exit). When defining the boundary with the circle (ID = 1000), a boundary is defined that is within the circle region

The settings shown for roads 1000 and 1001 in that figure are valid. (The settings shown for road 1002 were modified to show more options for boundaries based on station.)


[CarSim_Data] Road Boundary: New Road ID; { Roundabout D=38m } Circle in Roundabout

Function type: Step interpolation, flat-line extrapolation

Custom settings

Station (m), New road ID

	X Axis	Y Axis
1	0	1001
2	25	1002
3	50	1003
4	75	1004

Figure 49. New Road ID as a function of station for the roundabout circle.

Events and Trailers

When a VS Solver initializes, it iterates some calculations to estimate the initial vehicle location and orientation. To do so, it calculates the locations of all tire contact points.

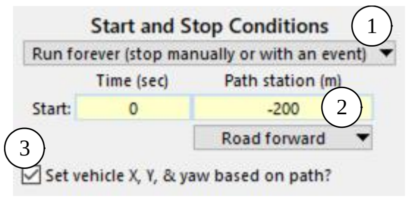
The initialization is done just before starting the simulation. If a VS Event is triggered during the simulation, the full initialization may be repeated, depending on which parameters were modified when reading Parsfiles when the Event is triggered.

When working with single-unit vehicle models (i.e., all BikeSim models, and CarSim and TruckSim models without trailers), the locations of points of interest used for the initialization during an Event are close to the locations when the Event was triggered. However, when the vehicle has one or more trailers, the initialization assumes zero articulation angles. In most cases, the results of the vehicle initialization are not used (the current state of the vehicle position and angles is maintained). However, if any of the points or interest are outside the road boundaries, the simulation might quit or generate an error.

If a set of connected roads will be used in simulations that involve trailers and VS Events, be sure that the boundaries are valid throughout the network. For example, in setting up boundaries, you might assume that the spaces between the lanes of the roundabout legs (see Figure 47, page 61) will never be used because realistic vehicle maneuvers would never put the tires in those regions. However, when simulating a vehicle with a trailer, it is possible to trigger Events where the trailer wheels would be in the gap when the articulation angle is zero during the processing of an Event.

Initial Vehicle Location

BikeSim, CarSim, and TruckSim support several options for setting the vehicle state before the run starts (via the **Procedures** screen) and making adjustments after the run is in progress (via the **Events** screen). On both screens, a checkbox is associated with setting the vehicle X, Y, and Yaw state variables (3, Figure 50). The relevant controls are described below.



Start and Stop Conditions

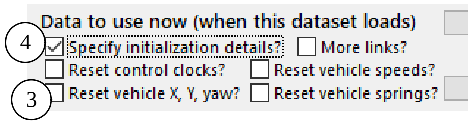
Run forever (stop manually or with an event)

	Time (sec)	Path station (m)
Start:	0	-200

Road forward

[✓] Set vehicle X, Y, & yaw based on path?

a. Procedure screen



Data to use now (when this dataset loads)

[✓] Specify initialization details? ☐ More links?

☐ Reset control clocks? ☐ Reset vehicle speeds?

☐ Reset vehicle X, Y, yaw? ☐ Reset vehicle springs?

b. Events screen

Figure 50. Setting vehicle location on the Procedures and Events screens.

(1) This drop-down control on the **Procedures** screen has options for setting the start and stop conditions. Most cause a yellow field to be shown for specifying the initial station of the vehicle (2) if the checkbox (3) is checked.

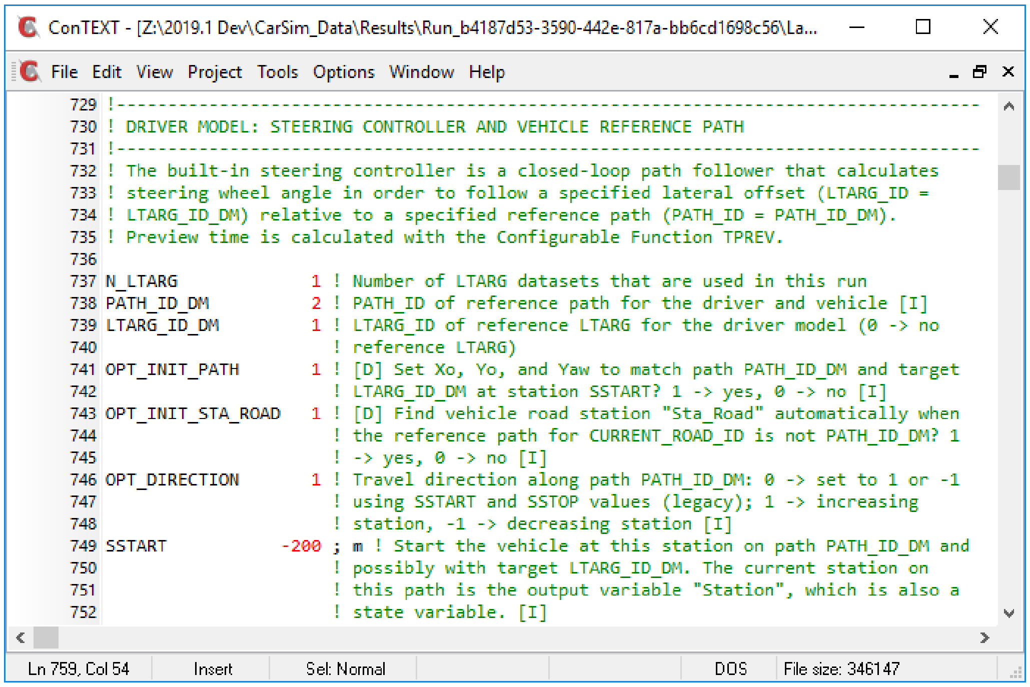
(2) Initial station of the vehicle (`SSTART`) on the path whose `PATH_ID` matches the parameter `PATH_ID_DM`.

(3) Option to set the vehicle X, Y, and Yaw based on a path and station, as described in the following subsection.

(4) Checkbox on the **Events** screen to show reset checkboxes, including (3).

Set Vehicle X, Y, and Yaw Based on Path

When the simulation involves a path used for a driver model or to define a VS Road, the vehicle is typically started with specified S and L coordinates for a path of interest. Recall that the Driver Model section of the Echo file lists parameters that identify the Reference Path for the vehicle and other information used to initialize the vehicle location (Figure 51).



ConTEXT - [Z:\2019.1 Dev\CarSim_Data\Results\Run_b4187d53-3590-442e-817a-bb6cd1698c56\La...

```
!--------------------------------------------------------------------------------------
! DRIVER MODEL: STEERING CONTROLLER AND VEHICLE REFERENCE PATH
!--------------------------------------------------------------------------------------
! The built-in steering controller is a closed-loop path follower that calculates
! steering wheel angle in order to follow a specified lateral offset (LTARG_ID =
! LTARG_ID_DM) relative to a specified reference path (PATH_ID = PATH_ID_DM).
! Preview time is calculated with the Configurable Function TPREV.

N_LTARG             1 ! Number of LTARG datasets that are used in this run
PATH_ID_DM          2 ! PATH_ID of reference path for the driver and vehicle [I]
LTARG_ID_DM         1 ! LTARG_ID of reference LTARG for the driver model (0 -> no
                      ! reference LTARG)
OPT_INIT_PATH       1 ! [D] Set Xo, Yo, and Yaw to match path PATH_ID_DM and target
                      ! LTARG_ID_DM at station SSTART? 1 -> yes, 0 -> no [I]
OPT_INIT_STA_ROAD   1 ! [D] Find vehicle road station "Sta_Road" automatically when
                      ! the reference path for CURRENT_ROAD_ID is not PATH_ID_DM? 1
                      ! -> yes, 0 -> no [I]
OPT_DIRECTION       1 ! Travel direction along path PATH_ID_DM: 0 -> set to 1 or -1
                      ! using SSTART and SSTOP values (legacy); 1 -> increasing
                      ! station, -1 -> decreasing station [I]
SSTART           -200 ; m ! Start the vehicle at this station on path PATH_ID_DM and
                      ! possibly with target LTARG_ID_DM. The current station on
                      ! this path is the output variable "Station", which is also a
                      ! state variable. [I]
```

Ln 759, Col 54 | Insert | Sel: Normal | DOS | File size: 346147

Figure 51. The Driver Model section of the Echo file shows initialization related to paths.

The Reference Path for the driver model and vehicle is specified with the path whose PATH_ID parameter matches PATH_ID_DM (line 738). An LTARG dataset may also be specified as the dataset whose LTARG_ID matches LTARG_ID_DM (line 739).

The path parameter OPT_INIT_PATH (checkbox ③, Figure 50, and line 741, Figure 51) may be used to specify that the vehicle will be initialized using PATH_ID_DM and possibly LTARG_ID. If OPT_INIT_PATH = 1, or if either the closed-loop steer controller or closed-loop speed controller is active, then the parameter OPT_DIRECTION (line 746) is used to set the direction in which the vehicle is moving along path PATH_ID_DM. Also, if OPT_INIT_PATH = 1, then the parameter SSTART (line 749) is used to set the initial S value for the vehicle Station on path PATH_ID_DM. The initial lateral coordinate is set by the LTARG dataset LTARG_ID_DM, unless LTARG_ID_DM = 0. In that case, the initial L coordinate is 0.

The horizontal position and orientation of the vehicle are defined with three state variables associated with the first sprung mass that can be set using either the output variable names Xo, Yo, and Yaw, or the state variable names SV_XO, SV_YO, and SV_YAW.

When OPT_INIT_PATH = 1, Xo, Yo, and Yaw are calculated automatically using path PATH_ID_DM, LTARG dataset LTARG_ID_DM, Station = SSTART, and OPT_DIRECTION.

When OPT_INIT_PATH = 0, Xo, Yo, and Yaw are not modified. They can be set explicitly in a miscellaneous field; otherwise they keep their default values, which are all 0.0.

Set Vehicle Station on VS Road Reference Path

If the road surface is defined with a set of VS Roads, the Reference Path ROAD_PATH_ID for CURRENT_ROAD_ID is the same as the vehicle/driver Reference Path PATH_ID_DM, then the initial station on the road for the vehicle (the state variable with the output name Sta_Road) is equal to the vehicle path state variable Station. However, when the driver path is not the road Reference Path, then Sta_Road and Station are typically different. In this case, the Sta_Road must be determined during initialization given the initial values of the coordinates Xo and Yo.

Note When VS Terrain is used, Sta_Road is never used and there is no potential ambiguity, and the rest of the this subsection is not applicable.

As noted earlier, it is possible that multiple sets of S-L coordinates may be mathematically valid for a given set of Xo and Yo coordinates (Figure 2, pager 5). As part of the initialization, the VS Math Model searches the Reference Path for CURRENT_ROAD_ID for possible valid values of Sta_Road. It chooses the station with the smallest magnitude for the corresponding L coordinate. That is, if finds the solution that is closest to the Reference Path.

An alternate option is to disable the automatic searching and specify an initial guess for Sta_Road. A parameter OPT_INIT_STA_ROAD (line 743, Figure 51) may be used to disable the search. (This parameter does not appear in the Echo file unless the Reference Path ROAD_PATH_ID for CURRENT_ROAD_ID differs from PATH_ID_DM.) If OPT_INIT_STA_ROAD = 0, then you can specify an estimate for Sta_Road in a miscellaneous yellow field.

Each time step, including the first, `Sta_Road` is calculated for the active road surface given `Xo` and `Yo` coordinates. This is also done for all tire contact points and all moving objects that are linked to a road surface. In simulations involving multiple road surfaces and boundaries, it is possible that different road surfaces are used for the different points. For this reason, each tire and moving object also includes a state variable with the Road ID used for that part, and another state variable for the road station.

Roughness Profiles

Roughness profiles allow realistic road roughness inputs to be used in simulated tests without requiring the extensive amounts of data needed to fully map a 3D surface, especially in cases where the vehicle is not following a straight path.

Profile Measurement

Roughness profiles are measured by highway agencies to evaluate the conditions of existing roads. Instrumented vehicles travel over the road to measure the pavement elevation in the left and right wheel tracks. The measurements are typically limited in bandwidth to include spatial frequencies (the inverse of wavelength, with units of cycle/m) that affect rigid-body vehicle dynamics in the range of 0.5 – 50 Hz at typical speeds. Vertical resolution is typically better than 0.5 mm, with sampling intervals of a few centimeters.

High-speed profiling devices usually do not have an accurate static reference for elevation; they obtain a reference elevation of each profiling measuring instrument by doubly-integrating a vertical accelerometer signal (with high-pass filtering to eliminate signal bias).

For more information about road profile measurement technology, see *The Little Book of Profiling*. (The Little Book was written in 1996 to provide an overview of road profiling technology for users of profiling instruments and the data obtained.) The PDF file for The Little Book is in the **Help** menu of the CarSim, TruckSim, and BikeSim browsers, and can be obtained via the Internet at:

http://deepblue.lib.umich.edu/handle/2027.42/21605

The Surface: Roughness Profiles Screen

In addition to the 3D surface defined either with a grid of X-Y coordinates or with a road reference line, the VS Math Models support a contribution to elevation based on surface roughness profiles:

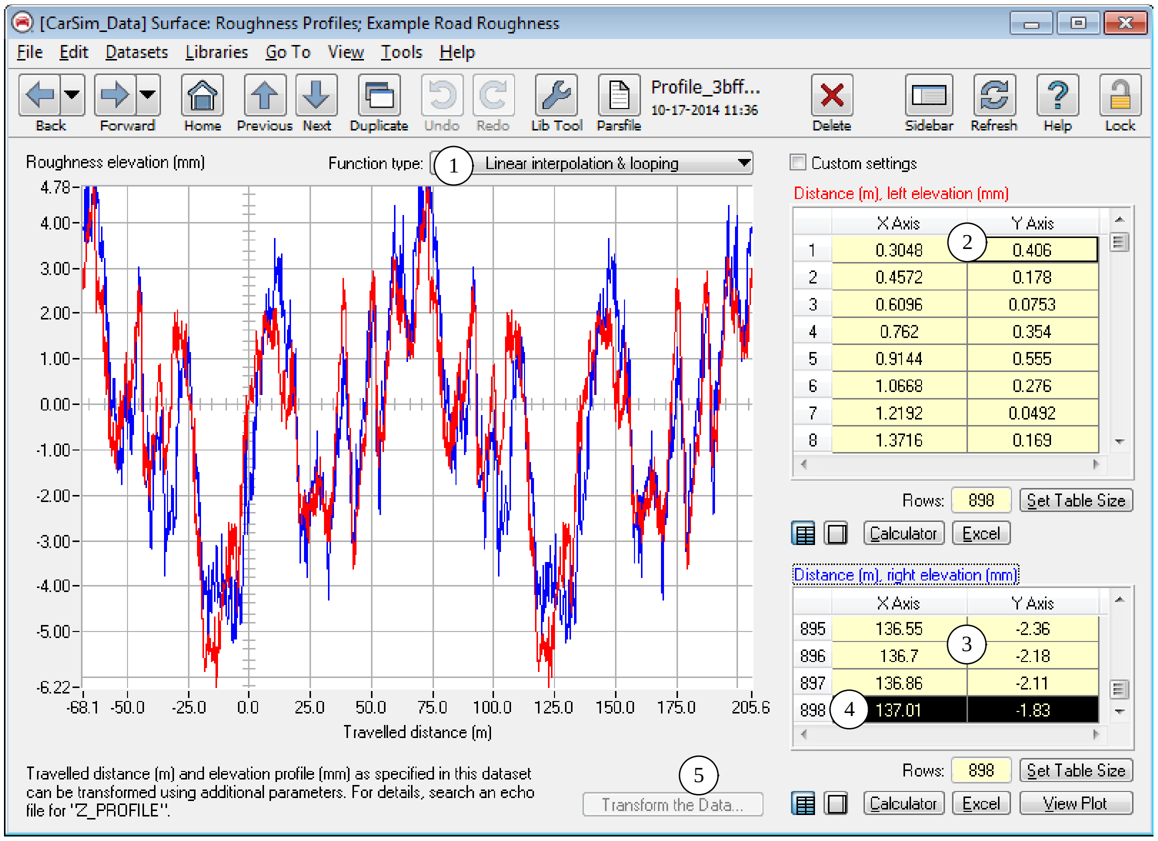
$$Z_t = Z_{fixed}(X, Y) + Z_{rp}(S_{td}) \tag{2}$$

where Z_t is the global Z coordinate where a tire contacts the ground, Z_{fixed} is the ground elevation defined as a fixed function of global X and Y coordinates (based on an internal transformation to road S and L coordinates), Z_{rp} is an elevation contribution from a roughness profile that follows the vehicle, and S_{td} is the distance travelled by the axle associated with the tire.

Along with the incremental contribution to elevation Z_t, the profile slope is added to the slope of the 3D surface in the X_T direction (X direction of the Tire axis system).

The roughness profiles are sometimes called “wandering profiles” because they follow the vehicle wherever it goes.

The Surface**: Roughness Profiles** screen in CarSim and TruckSim (Figure 52) have two tables of data: one for the left side tires and another for the right side tires. BikeSim has a single wandering profile.



Function type: Linear interpolation & looping

☐ Custom settings

Distance (m), left elevation (mm)

	X Axis	Y Axis
1	0.3048	0.406
2	0.4572	0.178
3	0.6096	0.0753
4	0.762	0.354
5	0.9144	0.555
6	1.0668	0.276
7	1.2192	0.0492
8	1.3716	0.169

Rows: 898

Distance (m), right elevation (mm)

	X Axis	Y Axis
895	136.55	-2.36
896	136.7	-2.18
897	136.86	-2.11
898	137.01	-1.83

Rows: 898

Travelled distance (m) and elevation profile (mm) as specified in this dataset can be transformed using additional parameters. For details, search an echo file for "Z_PROFILE".

Figure 52. The Surface: Roughness Profile screen.

The wandering profiles are assumed to have the longitudinal shapes shown in the plots, with elevation and corresponding longitudinal slope. In determining the ground geometry at a point where the tire makes contact, the elevation and longitudinal slope values from the profile contribute to the elevation and slope of the ground.

The value of travelled distance used with a roughness profile is obtained in the VS Math Model for the front axle by integrating the forward vehicle speed `SV_VX` to obtain the value of the state variable with the output name `Sta_Prof`. Values of travelled distance for all other axles are obtained by subtracting the static wheelbase of the trailing axle relative to the front axle. For example, if axle 2 is 2.8 m behind axle 1, then the distance used for obtaining Z and dZ/dS from a profile table is the distance calculated for the front axle, minus 2.8.

The wandering profile is an approximation used to provide roughness-induced vibrations without requiring the preparation of a full 3D surface dataset. It is not active in CarSim or TruckSim when using TNO MF-Tyre models or COSIN FTire.

User Settings

① Pull-down control to specify the table interpolation. Three options are provided: **Constant**; **Linear interpolation & looping**; and **Linear interpolation, flat-line extrapolation**.

② Table for the roughness profile for wheels on the left side of the vehicle. The first column is the independent variable distance, and the second column specifies the elevation to be added to the road surface.

③ Table for the roughness profile for wheels on the right side of the vehicle. The first column is the independent variable, distance, and the second column specifies the elevation to be added to the road surface.

⑤ **Transform the Data** button. Use this button to modify tabular data to provide continuous roughness excitation as described in the next subsection. This button is only active when the table type ① is set to **Linear interpolation, flat-line extrapolation**. It is not visible if the table type is set to **Constant**.

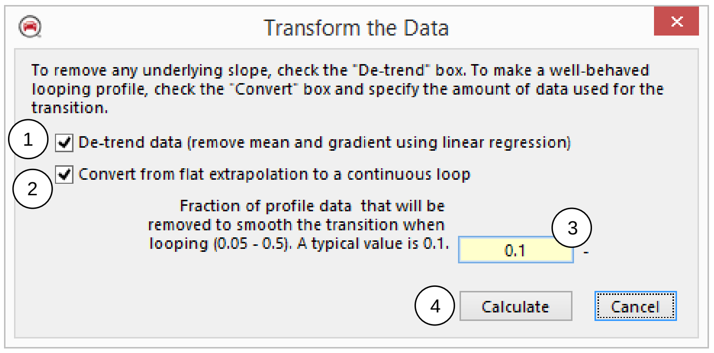
Looping Profiles

The roughness profiles can be made to loop endlessly, providing continuous excitation during a lengthy simulation. The looping is specified with the drop-down control ①, as shown in Figure 52.

In order for a looped profile to appear realistic, the roughness where the looping occurs (e.g., going from 137.01 m ④ to 0 m) should be statistically similar to the roughness everywhere else in the profile. If the first and last elevation values are not identical, the profile defines an abrupt discontinuity when a tire crosses the loop distance. Further, the variations just before and just after the looping should have the same approximate character as other parts of the profile.

This screen provides a calculator tool to transform profile data in the starting and ending regions to maintain roughness continuity. The method is based on years of research involving the measurement and analysis of road roughness profiles.

If the screen is currently showing tabular data with the calculation method set to **Linear interpolation, flat-line extrapolation**, then the button **Transform the Data** ⑤ becomes active. Clicking the button brings up a new window with settings to control the transformation (Figure 53).


Transform the Data

To remove any underlying slope, check the "De-trend" box. To make a well-behaved looping profile, check the "Convert" box and specify the amount of data used for the transition.

☑ De-trend data (remove mean and gradient using linear regression)

☑ Convert from flat extrapolation to a continuous loop

Fraction of profile data that will be removed to smooth the transition when looping (0.05 - 0.5). A typical value is 0.1. 0.1 -

Figure 53. Options to transform profile data to make looped profiles.

(1) Checkbox to de-trend the data by applying a linear regression to calculate the overall mean elevation Z and overall trend dZ/dS, where Z is the elevation in the table and S is the distance in the table. Both effects are removed, such that the remaining data have zero average elevation and zero trend.

(2) Checkbox to apply a cosine-taper filter to smooth the transition when looping. The conversion does the following:

1. Replaces the beginning of the profile for a range that is specified with a fraction (3) of the overall profile length
2. Shortens the profile by this amount.
3. Clicking the **Calculate** button (4) sets the Configurable Function on the **Surface: Roughness Profiles** screen from **Linear interpolation, flat-line extrapolation** to **Linear interpolation & looping** ((1), Figure 52).

(3) Fraction of profile data that will be removed to smooth the transition when looping. For example, a value of 0.1 (recommended) means that the beginning covers 1/10 of the total length and the profile is shortened by 1/10.

(4) Calculate button, used to apply the settings on this screen to the **Surface: Roughness Profiles** dataset (Figure 52).

The calculated values of the elevation points in the transition region are:

$$Znew_i = w\,Z_i + (1 - \mathrm{w})\,Z_{i+n-m} \tag{3}$$

where *n* is the number of points in the original table of profile data, *m* is the number of points in the transition region, $Znew_i$ is the new elevation value for point *i* of the profile, and *w* is a weighting factor calculated as a function of *i* that ranges from 0 (at point 1) to 1 (at point *m*). After filtering, the number of points in the profile is reduced from *n* to *n-m*.

Vehicle-Road Axis System

Standard vehicle dynamics terminology is specified in SAE J670 [1] and ISO 8855 [2]. Both standards define several Reference Frames with associated axis systems and coordinate systems.

Note A Reference Frame is a geometric environment in which all points remain fixed with respect to each other; an axis system is a set of orthogonal directions associated with *X*, *Y*, and *Z* axes; and a coordinate system is an axis system with an origin point.

Most of the motion variables defined in the standards make use of the following axis systems.

- **Earth-fixed** (X_E, Y_E, Z_E) — this system is fixed in the inertial Reference Frame. Z_E points up, in the direction opposite of gravity. X_E and Y_E define the *ground plane* that is level—perpendicular to gravity. The earth coordinate system is used to define global coordinates for points in the vehicle that locate the sprung mass, wheel centers, and other parts in a multibody model. The orientations of X_E and Y_E within the ground plane are arbitrary.

- **Vehicle** (X_V, Y_V, Z_V) — this system is fixed in the sprung mass of the vehicle. X_V points forward, Y_V points to the left, and Z_V points up. For vehicles that are essentially symmetric laterally, Y_V should be perpendicular to the plane of symmetry.
- **Intermediate** (X, Y, Z) — this is the system used to define major vehicle motion variables. Longitudinal velocity and acceleration are defined using the X-axis direction; lateral velocity and acceleration are defined using the Y-axis direction. These directions are completely defined by the direction of gravity ($Z = Z_E$) and the vehicle yaw angle.
- **Tire** (X_T, Y_T, Z_T) — a tire system exists for each tire and is used to define variables related to the contact of the tire with the ground. These directions are defined based on two direction vectors: the direction of the wheel spin axis (Y_W) and a direction perpendicular to a *road plane*, which is a local representation of the road surface in the region of the tire contact patch.

In addition to the above axis systems, VS Math Models include a **Road** axis system (X_R, Y_R, Z_R) for defining vehicle motion variables in the same way the **Intermediate** system is used. Instead of the main reference being gravity and a ground plane, the reference is the road normal, Z_R, which is perpendicular to the road surface. The point on the road surface used to determine the direction of Z_R is selected such that the Z_R axis passes through a designated point on the vehicle sprung mass. For CarSim and TruckSim, the vehicle reference point used to locate a point on the road surface is the aerodynamic reference point.

The X_R axis is perpendicular to Z_R and Y_V, and Y_R is perpendicular to X_R and Z_R. For example, Figure 54 shows both the Z_E and Z_R axes for a vehicle navigating a turn on a banked circular track. As the vehicle turns to its left, it rolls to the right (relative to the road normal), although the SAE/ISO roll angle is to the left due to the banking of the road. Thus, roll angle in the road axis system not only differs from the roll angle in the intermediate axis system — the two angles don't even have the same sign.

References

1. SAE International Surface Vehicle Recommended Practice, "Vehicle Dynamics Terminology," SAE Standard J670, Rev. 2008-01-24.
2. International Organization for Standardization, "Road vehicles — Vehicle dynamics and road-holding ability — Vocabulary," ISO Standard 8855, Rev. 2010.

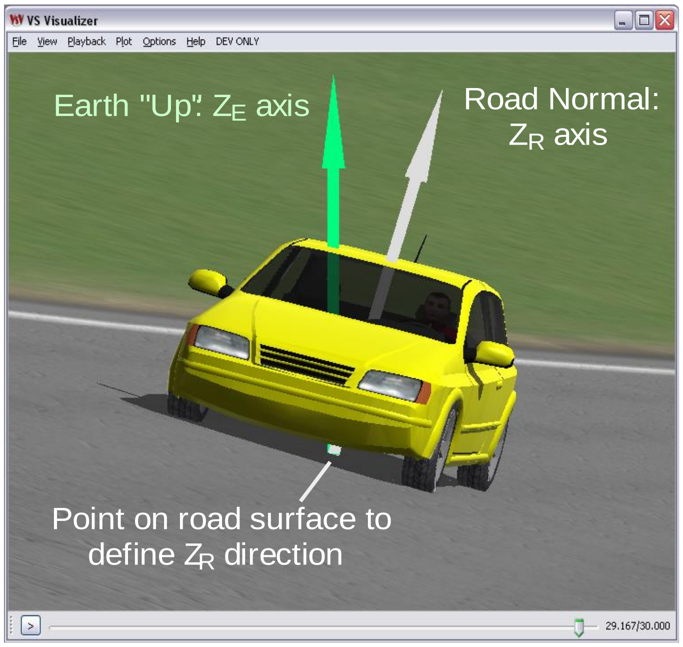

VS Visualizer
File View Playback Plot Options Help DEV ONLY
Earth "Up": Z_E axis
Road Normal: Z_R axis
Point on road surface to define Z_R direction
29.167/30.000

Figure 54. Comparison of Earth "up" and road normal for defining vehicle roll.

VS Commands for Paths and Roads

Paths and roads are added to the model automatically by the screen Parsfiles using VS Commands. The user IDs are also managed with VS Commands. Further, functions exist for obtaining information about paths and road surfaces that can be used in VS Commands.

Custom ID Numbers for Paths, Roads, and Configurable Functions

As noted earlier, some of the Configurable Functions related to paths are added as needed, as are paths and road surfaces. To help manage many optional datasets for these Configurable Functions, each includes an ID parameter that can be set to a unique integer value and used to help manage the use of the function.

Table 3 lists the keywords used for properties of Paths, Roads, and Configurable Functions that include custom ID parameters.

Table 3. Properties of Paths, Roads, and Configurable Functions with custom IDs.

Function or Object	No. of Datasets	Index	Custom ID
`LTARG`	`N_LTARG`	`ILTARG`	`LTARG_ID`
Path object	`NPATH`	`IPATH`	`PATH_ID`
Road object	`NROAD`	`IROAD`	`ROAD_ID`
`ROAD_DZ`	`NROAD_DZ`	`IROAD_DZ`	`ROAD_DZ_ID`
`SPEED_TARGET`	`N_SPEED_TARGET`	`ISPEED`	`SPEED_TARSET_ID`
XY table object	`NTAB_XY`	`ITAB_XY`	`XY_TABLE_ID`

For example, `N_LTARG` is the number of `LTARG` datasets in use, which is listed in the Echo file. When reading Parsfile data with `LTARG` parameters or tables, the context is defined by the current value of the index ILTARG. The custom ID parameter is LTARG_ID. That is the ID used to reference an LTARG dataset elsewhere in the math model. For example, the dataset used by the driver closed-loop steering model is LTARG_ID_DM, which means it uses the LTARG dataset with the specified LTARG_ID.

The table includes the function SPEED_TARGET, which is set with screens related to the driver or rider models. It is listed here because it is optional, and makes use of a custom ID. The custom ID is in turn used by driver and rider models, and to set the speed of moving objects.

The management of custom IDs is handled with VS Set Commands that are written in the Parsfiles for the Objects listed in Table 3. The VS Functions and Commands are listed in Table 4 for the six Functions and objects that support custom ID parameters.

Table 4. VS functions involving custom ID parameters.

Function or Object	**VS Command to set context**	**Get Index Function**
LTARG	SET_ILTARG_FOR_ID *uid*	GET_ILTARG(*uid*)
Path object	SET_IPATH_FOR_ID *uid*	GET_IPATH(*uid*)
Road object	SET_IROAD_FOR_ID *uid*	GET_IROAD(*uid*)
ROAD_DZ	SET_IROAD_DZ_FOR_ID *uid*	GET_IROAD_DZ(*uid*)
SPEED_TARGET	SET_ISPEED_FOR_ID *uid*	GET_ISPEED(*uid*)
XY table object	SET_ITAB_XY_FOR_ID *uid*	GET_ITAB_XY(*uid*)

Each of the Set functions has the same behavior, applied to a specific type of object. For example, the command SET_ILTARG_FOR_ID evaluates the argument *uid* (user ID). There are several possible outcomes:

1. If *uid* is zero, a new dataset is enabled by incrementing N_LTARG and the index ILTARG is set to N_LTARG. Until ILTARG is changed elsewhere, all LTARG-related information is applied to the LTARG dataset that was just added.
2. If *uid* ≥ 999, all N_LTARG of the existing LTARG_ID parameters and checked for a match with *uid*. If there is no match, a new dataset is enabled by incrementing N_LTARG, and ILTARG is set to N_LTARG.
3. If *uid* ≥ 999, the existing datasets are searched, and if all N_LTARG datasets are checked for an existing LTARG_ID parameter that matches *uid*. If found, ILTARG is set to the index of the dataset that matches. Until ILTARG is changed elsewhere, all LTARG-related information is applied to the LTARG dataset that was just identified.
4. If *uid* is not 0 and not ≥ 999, an error is generated and the simulation is not run.
5. If the command attempts to increase N_LTARG (for cases 1 or 2 above) and finds it is at the limit (currently 500 datasets), then an error is generated and the simulation is not run.

The Browser supports the GUI shown in **Error! Reference source not found.** by writing the appropriate SET function (e.g., SET_ILTARG_FOR_ID) with a value 0 for the automatic numbering, or the contents of the adjacent yellow field for a user-defined custom ID.

The GET functions from Table 4 provide the internal index as a value returned by the function. This is necessary to use a Configurable Functions based on custom ID. For example, to calculate L for the `LTARG` function, for the dataset with `LTARG_ID` = *uid*, use the form:

```
    LTARG(0,S,GET_ILTARG(uid))
```

Define New Paths, XY Tables, and Roads

The screens described in this document add new paths and roads using the Set commands described in the previous section.

Another option exists to define multiple datasets with a single `DEFINE_` command (Table 5). These commands are written automatically in Echo files to summarize in one command how many Paths, Roads, and XY tables were created for a simulation (at the time the Echo file was written).

Table 5. VS Define Commands for paths, roads, XY tables, and VS Terrain.

Function	**Description**
`DEFINE_PATHS` *n*	Define *n* new Reference Paths
`DEFINE_ROADS` *n*	Define *n* new Road Surfaces
`DEFINE_XY_TABLES` *n*	Define *n* new XY tables
`VS_TERRAIN_FILE`	Define the terrain with a `vsterrain` file

The `VS_TERRAIN_FILE` command is applied automatically when using a dataset from the **Scene: External Import** library, although it may also be employed from a miscellaneous field as is done with other VS Commands.

The `VS_TERRAIN_FILE` command and the `DEFINE_ROADS` commands are mutually exclusive, as described below.

DEFINE_PATHS

```
    DEFINE_PATHS n
```

This command defines *n* new reference paths. The number of paths currently in the model is assigned to the system parameter `NPATH`. Each path has a set of parameters plus a sequence of up to 100 contiguous segments. The path parameters are described in the document *Paths and Road Surfaces*, available from the **Help** menu.

All of the math model parameters that apply for a path are indexed to the path number as indicated by the current value of the system index parameter `IPATH`. All of the GUI screens that define new paths include a statement assign `IPATH` to the last path that was just defined (`IPATH` = `NPATH`).

A simulation will always include at least one reference path. If one is not included in the user inputs, then one is created automatically when the model initializes. In this case, the path is a straight line going along the global X-axis.

The first parameter for each path is `PATH_ID`. When the path is defined, this parameter is assigned to the index of the path (`PATH_ID` = `IPATH`). When a path is defined from within a VS Browser, the description for `PATH_ID` is set to the title of the dataset using the `SET_DESCRIPTION` command. (This is done to provide documentation in the Echo file if there are multiple paths used in the simulation.)

DEFINE_ROADS

```
DEFINE_ROADS n
```

This command defines *n* new road surfaces. The number of roads currently in the model is assigned to the system parameter `NROAD`. Each road has a set of parameters, plus a link to a Path, plus some associated Configurable Functions (Table 6), plus some optional links to other Configurable Functions. The road parameters and Configurable Functions are described in the document *Paths and Road Surfaces*, available from the **Help** menu.

Table 6. Configurable Functions that support Roads, with primary index IROAD.

Function	**Description**	**Argument(s)**	**2nd Index**
`MU_ROAD`	Friction of road surface	S, L	
`ROAD_DZ`	Component of Z	S, L	
`ROAD_L_BOUNDARY`	L coordinate of surface boundary	S	`ISIDE`
`ROAD_NEW_ID`	ID of next road surface across boundary	S	`ISIDE`
`ROAD_NEW_S`	Station of next road at boundary	S	`ISIDE`
`ROAD_ZS`	Component of Z	S	

All of the math model parameters that apply for a road are indexed to the road number as indicated by the current value of the system index parameter `IROAD`. The only screen in the GUI that defines a new road is **Road: 3D Surface (All Properties)**. When a new road is defined, the associated Parsfile assigned `IROAD` to the road that was just created (`IROAD` = `NROAD`).

Each of the Configurable Functions listed in Table 6 also use the `IROAD` index to identify a dataset. The three functions related to boundaries have a second index `ISIDE` to indicate which side of the surface (minimum L or maximum L) the dataset is used for.

The `VS_TERRAIN_FILE` command and the `DEFINE_ROADS` commands are mutually exclusive. If the `VS_TERRAIN_FILE` command has been applied, then any attempt to use the `DEFINE_ROADS` command will generate an error and stop the run.

On the other hand, if the `VS_TERRAIN_FILE` command has not been used, then the simulation will always include at least one road. If no road is included in the user inputs, then one is created automatically when the model initializes. In this case, the path for the road is a straight line going along the global X-axis, and the road is flat and level.

DEFINE_XY_TABLES

```
DEFINE_XY_TABLES n
```

This command defines *n* new XY tables for use in building paths. The number of XY tables currently in the model is assigned to the system parameter `NTAB_XY`.

The XY table is similar in appearance to a Configurable Function, but is handled different internally. Rather calculating Y as a function of X, it generates internal values of station S and then uses spline interpolation to calculate both X and Y as functions of S. For more information about these tables, please see the document *Paths and Road Surfaces*, available from the **Help** menu.

Each XY Table has two keywords: `XY_TABLE_ID` and `SEGMENT_XY_TABLE`. Both are indexed to the road number as indicated by the current value of the system index parameter `ITAB_XY`. When the XY table is defined, the `XY_TABLE_ID` parameter is assigned to the number of the table (`XY_TABLE_ID = NTAB_XY`). When a table is defined from within a VS Browser, the description for `XY_TABLE_ID` is set to the title of the dataset using the `SET_DESCRIPTION` command. (This is done to provide documentation in the Echo file if there are multiple paths used in the simulation.)

Any XY tables that were defined for a simulation are listed in the Echo file in their own section, after the Configurable Functions.

VS_TERRAIN_FILE

`VS_TERRAIN_FILE` *pathname*

This command installs a 3D ground surface as defined in the `vsterrain` file that identified with *pathname*. The parameter `CURRENT_ROAD_ID` is set to a value of 1 and is locked (it cannot be changed). Any existing road surfaces are cleared, and any future usage of the `DEFINE_ROADS` command will generate an error message.

The VS Terrain option is described in the *VS Terrain* and *VS Scene Builder* tech memos and the *Scene External Import* screen document.

VS Command to Initialize Station

As described by the previous section surrounding Figure 2, multiple S-L coordinates might be found when converting from X-Y coordinates. If an Event assigns a new value to `PATH_ID_DM`, it is possible for the solver to misidentify the station and lateral of the new path.

On option is to explicitly set the station for the new path, using the state variable `SV_STATION`, if the Event is set up so the station on the new path is known.

Another option is to use the command `INIT_CURRENT_PATH_SWEEP` to remedy the problem. This command will instruct the solver to search the entire path from end to end and assign the `SV_STATION` state variable a value that is as close as possible to the vehicle's current X and Y global coordinate.

VS Command Functions to Access Path and Road Information

Accessing information from a Path

Table 7 lists VS functions to access information about an arbitrary path. All of these functions include a user ID (*uid*). Those that involve coordinate transformations also include a query ID (*qid*), which is an integer between 1 and 99, inclusive. (The argument is rounded internally to the nearest integer.) Query ID numbers should be used to track a single point of interest moving along one or more paths.

Table 7. VS functions for interacting with a reference path using the User ID.

Function	**Description**
`PATH_SSTART_ID(`*uid*`)`	Starting station value of a reference path.
`PATH_SSTOP_ID(`*uid*`)`	Ending station value of a reference path.
`PATH_LENGTH_ID(`*uid*`)`	Length of a reference path.
`PATH_IS_LOOPED_ID(`*uid*`)`	Non-zero if the reference path is a loop.
`PATH_S_ID(`*x*, *y*, *uid*, *qid*`)`	The local S-L coordinates of the point (*x*,*y*).
`PATH_L_ID(`*x*, *y*, *uid*, *qid*`)`	
`PATH_X_ID(`*s*, *l*, *uid*, *qid*`)`	The global X-Y coordinates of a point (*s*,*l*) on the reference path.
`PATH_Y_ID(`*s*, *l*, *uid*, *qid*`)`	
`PATH_DXDS_ID(`*s*, *l*, *uid*, *qid*`)`	The gradients of the reference path at the point (*s*,*l*).
`PATH_DYDS_ID(`*s*, *l*, *uid*, *qid*`)`	
`PATH_DXDL_ID(`*s*, *l*, *uid*, *qid*`)`	
`PATH_DYDL_ID(`*s*, *l*, *uid*, *qid*`)`	
`PATH_CURV_ID(`*s*, *l*, *uid*, *qid*`)`	The curvature of the reference path at the point (*s*,*l*).
`PATH_YAW_ID(`*s*, *uid*, *uid_ltarg*, *qid*`)`	The instant yaw (heading) of a reference path at station *s*, including a possible effect of LTARG ID *uid_ltarg*

Each road uses a reference path to define the S-L coordinate system that in turn is used to define elevation and friction properties, and boundaries that might connect to other roads.

The last function in the table provides the instant yaw for a path for a given station, plus any additional effect on yaw from a linked `LTARG` dataset. If there is no `LTARG` dataset of interest, specify 0 for *uid_ltarg*.

Assessing information from a Road Surface

For a given road `IROAD`, the user ID of the reference path is specified with the parameter `ROAD_PATH_ID`. One way to transform between road S-L coordinates and global X-Y coordinates is to find the path and then use the functions from Table 7. However, as a shortcut, similar functions exist that use the road user ID (`ROAD_ID`) and automatically perform transformations using the path for that road (Table 8).

If a `vsterrain` file has been loaded with the `VS_TERRAIN_FILE` command, then the functions in Table 8 that give properties of a road reference line (`ROAD_SSTART_ID`, `ROAD_SSTOP_ID`, `ROAD_LENGTH_ID`, `ROAD_IS_LOOPED_ID`, and `ROAD_CURV_ID`) will all return 0.

The other functions in Table 8 work using the definition that S = X and L = Y (X and Y are global coordinates).

Table 8. VS functions for interacting with a road using User ID.

Function	**Description**
`ROAD_SSTART_ID(`*uid*`)`	Starting station value of the reference path for a road.
`ROAD_SSTOP_ID(`*uid*`)`	Ending station value of the reference path for a road.
`ROAD_LENGTH_ID(`*uid*`)`	Length of a road.
`ROAD_IS_LOOPED_ID(`*uid*`)`	Non-zero if the road is a loop.
`ROAD_S_ID(`*x, y, uid, qid*`)`	The local S-L coordinates of the point (*x*,*y*).
`ROAD_L_ID(`*x, y, uid, qid*`)`	
`ROAD_X_ID(`*s, l, uid, qid*`)`	The global X-Y coordinates of a point (*s*,*l*) on the reference path of a road.
`ROAD_Y_ID(`*s, l, uid, qid*`)`	
`ROAD_DXDS_ID(`*s, l, uid, qid*`)`	The gradients of the reference path of a road at the point (*s*,*l*).
`ROAD_DYDS_ID(`*s, l, uid, qid*`)`	
`ROAD_DXDL_ID(`*s, l, uid, qid*`)`	
`ROAD_DYDL_ID(`*s, l, uid, qid*`)`	
`ROAD_CURV_ID(`*s, l, uid, qid*`)`	The curvature of the ref. path of a road at the point (*s*,*l*).
`ROAD_Z_ID(`*s, l, uid, qid*`)`	The elevation of the road at a point (*s*,*l*).
`ROAD_DZDS_ID(`*s, l, uid, qid*`)`	The gradients of road elevation at a point (*s*,*l*).
`ROAD_DZDL_ID(`*s, l, uid, qid*`)`	

VS Command Functions directly accessible from Python

When running under Windows OS or non-RT Linux, VS Solvers can load Python and run Python code from within the model, as described in the *VS Command* reference manual and in the Technical Memo in *Extending a Model with Embedded Python.*

Table 9 lists VS Command functions related to paths and roads that can be applied by embedded Python.

Deprecated Path and Road VS Functions

Prior to CarSim 9.0 (2014), a simulation had one road and up to two reference paths:

1. A road reference path defined the road S-L coordinate system used to describe 3D ground geometry and friction.
2. A driver reference path was used by the built-in driver model that controls speed and steering (these commands include `PATH` in the name). The driver reference path might also have been used to define paths of moving objects.

In both cases, the path had a single segment defined with an XY table.

Table 10 lists functions that apply only to the road whose ID is specified by the system parameter `CURRENT_ROAD_ID`, and the path identified with the driver model parameter `PATH_DM_ID` (CarSim and TruckSim) or `OPT_RIDER` (BikeSim).

Table 9. VS Command Functions accessible with the Embedded Python Utility.

Command	**Description or VS Function**
`vs.path_sstart_id(uid)`	`PATH_SSTART_ID`(uid)
`vs.path_sstop_id(uid)`	`PATH_SSTOP_ID`(uid)
`vs.path_length_id(uid)`	`PATH_LENGTH_ID`(uid)
`vs.path_is_looped_id(uid)`	`PATH_IS_LOOPED_ID`(uid)
`vs.path_s_id(x, y, uid, qid)`	`PATH_S_ID`(x, y, uid, qid)
`vs.path_l_id(x, y, uid, qid)`	`PATH_L_ID`(x, y, uid, qid)
`vs.path_x_id(s, l, uid, qid)`	`PATH_X_ID`(s, l, uid, qid)
`vs.path_y_id(s, l, uid, qid)`	`PATH_Y_ID`(s, l, uid, qid)
`vs.path_dxds_id(s, l, uid, qid)`	`PATH_DXDS_ID`(s, l, uid, qid)
`vs.path_dyds_id(s, l, uid, qid)`	`PATH_DYDS_ID`(s, l, uid, qid)
`vs.path_dxdl_id(s, l, uid, qid)`	`PATH_DXDL_ID`(s, l, uid, qid)
`vs.path_dydl_id(s, l, uid, qid)`	`PATH_DYDL_ID(`s, l, uid, qid)
`vs.path_curv_id(s, l, uid, qid)`	`PATH_CURV_ID`(s, l, uid, qid)
`vs.path_yaw_id(s, uid, uid_ltarg,qid)`	`PATH_YAW_ID`(s, uid, uid_ltarg, qid)
`vs.road_sstart_id(uid)`	`ROAD_SSTART_ID`(uid)
`vs.road_sstop_id(uid)`	`ROAD_SSTOP_ID`(uid)
`vs.road_length_id(uid)`	`ROAD_LENGTH_ID`(uid)
`vs.road_is_looped_id(uid)`	`ROAD_IS_LOOPED_ID`(uid)
`vs.road_s_id(x, y, uid, qid)`	`ROAD_S_ID`(x, y, uid, qid)
`vs.road_l_id(x, y, uid, qid)`	`ROAD_L_ID`(x, y, uid, qid)
`vs.road_x_id(s, l, uid, qid)`	`ROAD_X_ID`(s, l, uid, qid)
`vs.road_y_id(s, l, uid, qid)`	`ROAD_Y_ID`(s, l, uid, qid)
`vs.road_dxds_id(s, l, uid, qid)`	`ROAD_DXDS_ID`(s, l, uid, qid)
`vs.road_dyds_id(s, l, uid, qid)`	`ROAD_DYDS_ID`(s, l, uid, qid)
`vs.road_dxdl_id(s, l, uid, qid)`	`ROAD_DXDL_ID`(s, l, uid, qid)
`vs.road_dydl_id(s, l, uid, qid)`	`ROAD_DYDL_ID`(s, l, uid, qid)
`vs.road_curv_id(s, l, uid, qid)`	`ROAD_CURV_ID`(s, l, uid, qid)
`vs.road_z_id(s, l, uid, qid)`	`ROAD_Z_ID`(s, l, uid, qid)
`vs.road_dzds_id(s, l, uid, qid)`	`ROAD_DZDS_ID`(s, l, uid, qid)
`vs.road_dzdl_id(s, l, uid, qid)`	`ROAD_DZDL_ID`(s, l, uid, qid)
`vs.statement(key, buffer, stop)`	`VS_STATEMENT`(key, buffer, stop)

Note Mechanical Simulation does not recommend using the functions listed in Table 10. They exist now only to provide support for older datasets. The road functions do not work at all if the ground geometry is specified with VS Terrain.

Table 10. Deprecated VS functions for interacting with a road or driver reference path.

Function	**Description**
`PATH_CURV_I(`*s*, *qid*`)`	Curvature of reference path at station S for query ID *qid*.
`ROAD_CURV_I(`*s*, *qid*`)`	
`PATH_L_I(`*x*, *y*, *qid*`)`	Lateral offset L of point defined by X and Y coordinates, relative to reference path.
`ROAD_L(`*x*, *y*`)`	
`ROAD_L_I(`*x*, *y*, *qid*`)`	
`PATH_S_I(`*x*, *y*, *qid*`)`	Distance along reference path (station S) to point defined by X and Y coordinates.
`ROAD_S(`*x*, *y*`)`	
`ROAD_S_I(`*x*, *y*, *qid*`)`	
`PATH_X_I(`*s*, *qid*`)`	X coordinate at station S along reference path.
`ROAD_X(`*s*`)`	
`ROAD_X_I(`*s*, *qid*`)`	
`PATH_X_SL_I(`*s*, *l*, *qid*`)`	X coordinate of point defined by S and L coordinates.
`ROAD_X_SL_I(`*s*, *l*, *qid*`)`	
`PATH_Y_I(`*s*, *qid*`)`	Y coordinate at station S along reference path.
`ROAD_Y(`*s*`)`	
`ROAD_Y_I(`*s*, *qid*`)`	
`PATH_Y_SL_I(`*s*, *l*, *qid*`)`	Y coordinate of point defined by S and L coordinates.
`ROAD_Y_SL_I(`*s*, *l*, *qid*`)`	
`PATH_YAW_I(`*s*, *dir*, *qid*`)`	Heading (*yaw*) of reference path for direction *dir* = 1 or -1.
`ROAD_YAW(`*s*, *dir*`)`	
`ROAD_YAW_I(`*s*, *dir*, *qid*`)`	
`TARGET_HEADING(`*s*`)`	Target heading angle (relative to path) as function of S.
`TARGET_L(`*s*`)`	Target lateral position as function of S.
Note: The query ID *qid* can be set to a number from 1-99; the argument is rounded internally to the nearest integer. If there is no query argument (e.g., `ROAD_L`) then the number is set internally to 99.	

The functions listed in Table 10 with `PATH` in the name perform transformations related to the driver reference path. Often, the driver path and the road reference line are the same, which means S and L coordinates for the road are the same as the S and L coordinates for the driver path. It is also possible to have the 3D road enabled and have a separate path for the closed-loop driver model. In this case, S and L values for the functions `ROAD` functions differ from those used in the `PATH` functions.

The two functions `TARGET_HEADING` and `TARGET_L` refer to the target path for the internal driver model. The target path is defined relative to the driver reference path. (See the *Driver Controls* documentation for details on the target path and reference line.)

Table 11 lists more deprecated functions involving road geometry.

Table 11. Deprecated road geometry functions for VS symbolic expressions.

Function	**Description**
`ROAD_PITCH_SL_I`(*s*, *l*, *yaw*, *qid*)	Pitch angle of road at S and L coordinates with heading angle *yaw*, based on yaw-pitch-roll Euler angles.
`ROAD_ROLL_SL_I`(*s*, *l*, *yaw*, *qid*)	Roll angle of road at S and L coordinates with heading angle *yaw*, based on yaw-pitch-roll Euler angles.
`ROAD_Z`(*x*, *y*)	Z coordinate of point defined by X and Y coordinates.
`ROAD_Z_I`(*x*, *y*, *qid*)	
`ROAD_Z_SL_I`(*s*, *l*, *qid*)	Z coordinate of point defined by S and L coordinates. In this case, S and L are always for the road reference line.
Note: The query ID *qid* can be set to a number from 1-99; the argument is rounded internally to the nearest integer. If there is no query argument (e.g., `ROAD_L`) then the number is set internally to 99.	

Note Mechanical Simulation does not recommend using the functions listed in Table 11. They exist now only to provide support for older datasets. They do not work at all if the ground geometry is specified with VS Terrain.

The command `DEFINE_DZ_TABLES` was introduced when the `ROAD_DZ` Configurable Function was made optional for road surfaces. It is still functional, in support of older datasets. However, the preferred method for increasing the number of active `ROAD_DZ` Configurable Functions is to simply change the parameters `NROAD_DZ` and `IROAD_DZ`.

VS API Functions for Paths and Roads

The VehicleSim API includes a number of functions that and create and access paths and road surfaces. These can be applied by external programs that can dynamically load a DLL (or Linux SO) file and access the built-in API functions.

Functions to Obtain Information about Paths

The VS API includes functions to query these roads and paths directly using the User ID. The functions make use of the same instance numbers (between 1 and 99) used by the legacy road and path functions. As with many other VS API functions, the User ID is passed in as a double and rounded to the nearest integer internally.

Table 12 lists API functions for getting information about any existing path, identified with the path User ID.

Table 12. VS functions for interacting with a specified reference path.

Function	Description

`vs_path_sstart_id`
`vs_path_sstop_id`
`vs_path_length_id`
`vs_path_is_looped_id`
	Get a property from a specified path: sstart, sstop, length, and is_looped

`vs_road_curv_i`	
`vs_path_s_id` `vs_path_l_id`	Get path coordinates S or L for a point defined by X and Y coordinates
`vs_path_x_id` `vs_path_y_id`	Get global X and Y coordinates for a point on a specified path defined with S and L coordinates.

`vs_path_dxds_id`
`vs_path_dyds_id`
`vs_path_dxdl_id`
`vs_path_dydl_id`
	Get a partial derivative of X or Y with respect to S or L for a specified path, for a point defined with S and L coordinates.

`vs_path_yaw_id`	Get the yaw angle for a point on a path with a linked `LTARG` dataset, given S.
`vs_path_curv_id`	Get the curvature for a path at a given set of S-L coordinates.

vs_path_sstart_id
vs_path_sstop_id
vs_path_length_id
vs_path_is_looped_id

Declarations:

```
double vs_path_sstart_id(double user_id);
double vs_path_sstop_id double user_id);
double vs_path_length_id(double user_id);
double vs_path_is_looped_id(double user_id);
```

Return:

Each function returns the property implied by the name. These are: the starting and stopping station of the path, its length, and whether it is treated as a loop. A non-zero return value from `vs_path_is_looped_id` indicates the path is looped.

vs_path_s_id
vs_path_l_id

Declarations:

```
double vs_path_s_id(double x, double y, double user_id,
                    double inst);
double vs_path_l_id(double x, double y, double user_id,
                    double inst);
```

Return:

The S-L coordinates of a point with the given X-Y coordinates.

vs_path_x_id
vs_path_y_id

Declarations:

```
double vs_path_x_id(double s, double l, double user_id,
                    double inst);
double vs_path_y_id(double s, double l, double user_id,
                    double inst);
```

Return:

The X-Y coordinates of a point with the given S-L coordinates.

vs_path_dxds_id
vs_path_dyds_id
vs_path_dxdl_id
vs_path_dydl_id

Declarations:

```
double vs_path_dxds_id(double s, double l, double user_id,
                       double inst);
double vs_path_dyds_id(double s, double l, double user_id,
                       double inst);
double vs_path_dxdl_id(double s, double l, double user_id,
                       double inst);
double vs_path_dydl_id(double s, double l, double user_id,
                       double inst);
```

Return:

The gradients of a path at a point with the given S-L coordinates.

vs_path_yaw_id

Declaration:

```
double vs_path_curv_id(double s, double path_id,
                       double ltarg_id, double inst);
```

Return:

The yaw angle (heading) of the path with user ID `path_id`, plus a yaw contribution from an `LTARG` dataset with user ID `ltarg_id` . If there is no linked `LTARG` dataset, specify zero for `ltarg_id`.

vs_path_curv_id

Declaration:

```
double vs_path_curv_id(double s, double l, double user_id,
                       double inst);
```

Return:

The curvature of the path at the point with the given S-L: coordinates.

Functions to Obtain Information about Road Surfaces

Along with the functions provide information related to paths identified with User IDs, the API includes functions to provide information related to road surfaces identified with User IDs (Table 13). Most of these functions are similar in function to those described in the previous subsection, with the difference being that the use a road User ID to obtain information about the reference path of the road, rather than a path User ID. In addition, there are three functions that provide elevation information involving the Z coordinate of the road surface.

Table 13. VS functions for interacting with a specified road surface.

Function	Description

`vs_road_sstart_id`
`vs_road_sstop_id`
`vs_road_length_id`
`vs_road_is_looped_id`
	Get a property from the reference path for a specified road: sstart, sstop, length, and is_looped

`vs_road_s_id` `vs_road_l_id`	Get path coordinates S or L for the reference path for a specified road, for a point defined by X and Y coordinates
`vs_road_x_id` `vs_road_y_id`	Get global X and Y coordinates for a point on a reference path for a specified road, with S and L coordinates.

`vs_road_dxds_id`
`vs_road_dyds_id`
`vs_road_dxdl_id`
`vs_road_dydl_id`
	Get a partial derivative of X or Y with respect to S or L for a reference path for a specified road, for a point defined with S and L coordinates.

`vs_road_curv_id`	Get the curvature for a reference path for a specified road, at a given set of S-L coordinates.

`vs_road_z_id`
`vs_road_dzds_id`
`vs_road_dzdl_id`
	Get the Z coordinate and derivative of a specified road surface, given set of S-L coordinates.

vs_road_sstart_id
vs_road_sstop_id
vs_road_length_id
vs_road_is_looped_id

Declarations:

```
double vs_road_sstart_id(double user_id);
double vs_road_sstop_id(double user_id);
double vs_road_length_id(double user_id);
double vs_road_is_looped_id(double user_id);
```

Return:

Each function returns the property implied by the name. These are: the starting and stopping station of the path, its length, and whether it is treated as a loop. A non-zero return value from `vs_path_is_looped_id` indicates the road is looped.

vs_road_s_id
vs_road_l_id

Declarations:

```
double vs_road_s_id(double x, double y, double user_id,
                    double inst);
double vs_road_l_id(double x, double y, double user_id,
                    double inst);
```

Return:

The S-L coordinates of a point at the given X-Y coordinates on the road.

vs_road_x_id
vs_road_y_id

Declarations:

```
double vs_road_x_id(double s, double l, double user_id,
                    double inst);
double vs_road_y_id(double s, double l, double user_id,
                    double inst);
```

Return:

The X-Y coordinates of a point on the road with the given S-L coordinates.

vs_road_dxds_id
vs_road_dyds_id
vs_road_dxdl_id
vs_road_dydl_id

Declarations:

```
double vs_road_dxds_id(double s, double l, double user_id,
                       double inst);
double vs_road_dyds_id(double s, double l, double user_id,
                       double inst);
double vs_road_dxdl_id(double s, double l, double user_id,
                       double inst);
double vs_road_dydl_id(double s, double l, double user_id,
                       double inst);
```

Return:

The gradients of the road reference path at the point with the given S-L coordinates.

vs_road_curv_id

Declaration:

```
double vs_road_curv_id(double s, double l, double user_id,
                       double inst);
```

Return:

The curvature of the road at the point with the specified S-L coordinates.

vs_road_z_id
vs_road_dzds_id
vs_road_dzdl_id
Declarations:

```
double vs_road_z_id(double s, double l, double user_id,
                    double inst);
double vs_road_dzds_id(double s, double l, double user_id,
                       double inst);
double vs_road_dzdl_id(double s, double l, double user_id,
                       double inst);
```

Return:

The Z coordinate and derivatives of the road at the point with the specified S-L coordinates.

Deprecated API Functions for the Reference Path and Current Road

Older versions of the vehicle simulations included a single road surface and two reference paths: one to define the road and one to be used as a target by the driver model. Now that the simulations include multiple paths and roads, newer API functions can be used with any existing path or road.

Alert The functions documented in this section are no longer used at Mechanical Simulation, nor are the covered by internal testing.

Table 10 lists legacy API functions available to make use of the path used to locate the vehicle, specified with the parameter `PATH_ID_DM`, and the reference path for the current road surface as specified with the parameter `CURRENT_ROAD_ID`.

vs_get_road_start_stop
Declaration:

```
void vs_get_road_start_stop (double *start, double *stop);
```

Get the current start and stop stations in the path specified with the parameter `PATH_ID_DM` that is looped.

Note The name here is misleading; this function existed long before the other path-related functions and does not follow the newer naming convention.

vs_path_curv_i
vs_road_curv_i
Declarations:

```
double vs_path_curv_i(double s, double i);
double vs_road_curv_i(double s, double i);
```

Return:

Lateral curvature of reference path.

Table 14. Deprecated VS functions for interacting with the current road or driver reference path.

Function	**Description**
`vs_get_road_start_stop`	Get start and stop stations for looped driver reference path.
`vs_path_curv_i`	Curvature of reference path at station S for object i.
`vs_road_curv_i`	
`vs_path_l_i`	Lateral offset L of point defined by X and Y coordinates, relative to reference path.
`vs_road_l`	
`vs_road_l_i`	
`vs_path_s_i`	Distance along reference path (station S) to point defined by X and Y coordinates.
`vs_road_s`	
`vs_road_s_i`	
`vs_path_x_i`	X coordinate at station S along reference path.
`vs_road_x`	
`vs_road_x_i`	
`vs_path_x_sl_i`	X coordinate of point defined by S and L coordinates.
`vs_road_x_sl_i`	
`vs_path_y_i`	Y coordinate at station S along reference path.
`vs_road_y`	
`vs_road_y_i`	
`vs_path_y_sl_i`	Y coordinate of point defined by S and L coordinates.
`vs_road_y_sl_i`	
`vs_path_yaw_i`	Heading (yaw) of reference path for direction dir = 1 or -1.
`vs_road_yaw`	
`vs_road_yaw_i`	
`vs_s_loop`	Get station (S) as possibly limited in a looped reference path.
`vs_s_path_loop`	
`vs_target_heading`	Target heading angle (relative to path) as function of S.
`vs_target_l`	Target lateral position as function of S.

Curvature to the left is positive when travelling in the direction of increasing station.

The instance number *i* is rounded off internally to the nearest integer and should be between 1 and 99. The function `vs_path_curve_i` uses the driver reference path; the function `vs_road_curv_i` uses the road reference line.

`vs_path_l_i`
`vs_road_l`
`vs_road_l_i`

Declarations:

```
double vs_path_l_i(double x, double y, double i);
double vs_road_l(double x, double y);
double vs_road_l_i(double x, double y, double i);
```

Return:

Lateral position associated with a pair of X and Y values.

The instance number *i* is rounded off internally to the nearest integer and should be between 1 and 99. The functions `vs_road_l` and `vs_road_l_i` use the road reference line; the function `vs_path_l_i` uses the driver reference path.

vs_path_s_i
vs_road_s
vs_road_s_i

Declarations:

```
double vs_path_s_i(double x, double y, double i);
double vs_road_s(double x, double y);
double vs_road_s_i(double x, double y, double i);
```

Return:

Station associated with a pair of X and Y values.

The instance number *i* is rounded off internally to the nearest integer and should be between 1 and 99. The functions `vs_road_s` and `vs_road_s_i` use the road reference line; the function `vs_path_s_i` uses the driver reference path.

vs_path_x_i
vs_path_y_i
vs_road_x
vs_road_x_i
vs_road_y
vs_road_y_i

Declarations:

```
double vs_path_x_i(double s, double i);
double vs_path_y_i(double s, double i);
double vs_road_x(double s);
double vs_road_x_i(double s, double i);
double vs_road_y(double s);
double vs_road_y_i(double s, double i);
```

Return:

X or Y coordinate for station S on the reference line.

The object instance *i* is rounded off internally to the nearest integer and should be between 1 and 99. The functions `vs_path_x_i` and `vs_path_y_i` use the driver reference path; the others use the road reference line.

vs_path_x_sl_i
vs_path_y_sl_i
vs_road_x_sl_i
vs_road_y_sl_i

Declarations:

```
double vs_path_x_sl_i(double s, double l, double i);
double vs_path_y_sl_i(double s, double l, double i);
double vs_road_x_sl_i(double s, double l, double i);
double vs_road_y_sl_i(double s, double l, double i);
```

Return:

X or Y coordinate for station S and lateral coordinate L.

The object instance *i* is rounded off internally to the nearest integer and should be between 1 and 99. The functions `vs_path_x_sl_i` and `vs_path_y_sl_i` use the driver reference path; the others use the road reference line.

vs_path_yaw_i
vs_road_yaw
vs_road_yaw_i

Declarations:

```
double vs_path_yaw(double s, double direction);
double vs_road_yaw(double s, double direction);
double vs_road_yaw_i(double s, double direction, double i);
```

Return:

Yaw (heading) angle for station S on the reference line.

Set direction to 1.0 for the direction of increasing S, or -1.0 for the direction of decreasing S. The instance number *i* is rounded off internally to the nearest integer and should be between 1 and 99. The function `vs_path_yaw_i` uses the driver reference path; the others use the road reference line.

vs_s_loop
vs_s_path_loop

Declarations:

```
double vs_s_loop(double s);
double vs_s_path_loop(double s);
```

Return:

Station within the value range for a looped road, such as a test track or racetrack.

The function `vs_s_path_loop` uses the driver reference path; the returned value will be within the limits obtained with the function `vs_get_road_start_stop`. The function `vs_s_loop` uses the road reference line.

vs_target_heading

Declaration:

```
double vs_target_heading(double s);
```

Return the heading angle of a target path (radians), relative to the driver reference path, as a function of station. The internal driver model uses this target path.

This function is also available in VS symbolic equations via the symbolic function `TARGET_HEADING`.

vs_target_l

Declaration:

```
double vs_target_l(double s);
```

Return the lateral position of a target path, relative to the driver reference path, as a function of station. The internal driver model uses this target path.

Road 3D Surface Function Definitions

3D ground geometry is defined such that elevation Z and friction μ are specified in terms of S and L coordinates for a road reference line. Table 15 lists the functions that give information about the current road surface as specified with the parameter `CURRENT_ROAD_ID`.

Table 15. Deprecated functions in the VS API to access the 3D surface properties.

Function	**Description**
`vs_get_dzds_dzdl`	Get ground slopes at specified S and L values.
`vs_get_dzds_dzdl_i`	
`vs_get_road_contact`	Get Z, slopes, and friction from X and Y.
`vs_get_road_contact_sl`	Get Z, slopes, and friction from S and L.
`vs_get_road_xyz`	Get X, Y, and Z coordinates from S and L.
`vs_road_pitch_sl_i`	Get road pitch from S, L, Yaw.
`vs_road_roll_sl_i`	Get road roll from S, L, Yaw.
`vs_road_z`	Get road Z coordinate from X and Y.
`vs_road_z_i`	
`vs_road_z_sl_i`	Get road Z coordinate from S and L.

vs_get_dzds_dzdl
vs_get_dzds_dzdl_i

Declarations:

```
void vs_get_dzds_dzdl (double s, double l, double *dzds,
                       double *dzdl);
void vs_get_dzds_dzdl_i (double s, double l, double *dzds,
                         double *dzdl, double i);
```

Get the current slopes of road elevation (Z) with respect to S and L (`dzds` and `dzdl`, respectively). The instance number *i* is rounded off internally to the nearest integer and should be between 1 and 99.

vs_get_road_contact
vs_get_road_contact_sl

Declarations:

```
void vs_get_road_contact (double y, double x, int inst,
      double *z, double *dzdy, double *dzdx, double *mu);
```

```
void vs_get_road_contact_sl (double s, double l, int inst,
      double *z, double *dzdy, double *dzdx, double *mu);
```

Get information about a point on the road (typically a center of tire contact), given the X and Y coordinates of the point or the S and L coordinates. The argument *inst* is an instance number and should be given a value ranging from 201 to 299. Repeated calls to the function with the same instance will give results in the same proximity, which can be important with complicated road geometries where there can be multiple solutions (S and L values) for a given point defined by X and Y coordinates.

These functions provide values for the pointer arguments, where `z` is the Z coordinate of the point, the arguments *dzdx* and *dzdy* are the slopes of the road surface with respect to X and Y, respectively, and *mu* is the coefficient of tire/ground friction.

vs_get_road_xyz

Declaration:

```
void vs_get_road_xyz (double s, double l, double *x,
                      double *y, double *z);
```

Get the X, Y, and Z coordinates from a pair of S and L values (S and L apply to the road reference line).

vs_road_pitch_sl_i
vs_road_roll_sl_i

Declarations:

```
double vs_road_pitch_sl_i(double s, double l, double yaw,
                          double i);
double vs_road_roll_sl_i(double s, double l, double yaw,
                         double i);
```

Return:

Pitch or roll angle at a point on the road defined with a pair of S and L values for the road reference line and a global yaw angle.

The instance number *i* is rounded off internally to the nearest integer and should be between 1 and 99.

vs_road_z
vs_road_z_i
vs_road_z_sl_i

Declarations:

```
double vs_road_z(double x, double y);
double vs_road_z_i(double x, double y, double i);
double vs_road_z_sl_i(double s, double l, double i);
```

Return:

Z coordinate for a pair of X and Y values or a pair of S and L values.

The instance number *i* is rounded off internally to the nearest integer and should be between 1 and 99.

Troubleshooting

This section includes troubleshooting steps and recommendations related to issues that might arise when setting up roads and paths, including data that might be created from physically measured data such as GPS.

S and L Errors (Station and Lateral Position)

As noted in the section Coordinate Transformations (page 4), a significant complication arises when there are multiple pairs of S-L coordinates that would be valid for a single X-Y point of interest. When running a simulation, this issue can present itself as an error message indicating that there are multiple valid Station and Lateral Position coordinates. This can happen for a variety of reasons, including:

- The vehicle has been driven very far from the road centerline.
- The road has tight turns and/or many turns very close to one another.
- The road data's origin is very far from the VS Solver's road origin of (0, 0).
- The road geometry has vertical sections, as can happen when a section of physically measured road data is extrapolated.
- Long vehicles (i.e., a straight truck or trailer) negotiate a small radius turn and the vehicle off-tracks sufficiently such that it approaches the geometric center of the turn.

If vehicle has been driven far from the road centerline, the solution is to prevent this from happening, either by changing the Driver Controls or modifying the road data. Road Boundaries or Events can also be used to monitor for and prevent such a condition from occurring.

For road geometry-related S-L errors, make sure the geometry is reasonable — no vertical sections, no sharp corners, and turns cannot be physically impossible for the vehicle to navigate. If the road geometry is correct but S-L errors are still occurring, one option is to specify a separate path for the Driver Model. The path would be set on top of the road and may even follow the centerline

Origin of the Road is Very Far from (0, 0)

CarSim, TruckSim, and BikeSim support options that allow the origin of the roads and paths to be set to a location other than the default, (0, 0) meters. When this new position is thousands of meters away from (0, 0), however, various issues can arise, ranging from S-L errors to poorly rendered road animation shapes.

For physically measured road and path data (i.e., GPS) such as this, the recommendation is to reset its origin to (0, 0) meters. This can be accomplished either before it's copied into the VS Browser, or the built-in Calculator can be used to reset the data (e.g., see the bottom of Figure 13).

Auto-Generated Road Shapes Render Poorly when X-Y Coordinates are Huge

Some end-users working with road data that is many hundreds or thousands of meters long have reported that the road and any linked **Road: Animator Repeated Objects** (e.g., trees, signs, etc.) render poorly — the animation is choppy and appears to have missing data.

This is a subset of the section above regarding the origin of roads being very far from (0, 0) meters, and is expected behavior with floating point precision loss when translating between thousands or millions of meters. As noted above, the solution is the same: translate the origin of the road data to the VS Solver default location of (0, 0) meters.

Road: X-Y-Z Coordinates of Edges screen: Errors with Upper vs. Lower Table

As noted in an earlier section of the document (page 50), the **Road: X-Y-Z Coordinates of Edges** screen allows end-users to enter data representing the coordinates of the edges of the road, from which the VS Solver will linearly interpolate between these points to create the road surface on which the vehicle will drive. Unlike the other road libraries, the **Road: X-Y-Z Coordinates of Edges** screen allows the end-user to select as the road's X-Y reference path either the **Upper Table** or the **Lower Table**; i.e., either the left edge or the right edge of the road serves as the road reference path, not the centerline.

Instances exist in which the VS Solver will run as expected when the **Lower Table** is chosen as the reference path but will produce S-L errors when the **Upper Table** is chosen and vice versa. If this happens, the most common cause is too much curvature exists in the data and the simulation fails to locate the vehicle's S-L coordinates for a given set of X-Y coordinates. When this happens, the recommendations are as follows:

- Reduce the curvature implied by the data.
- If reducing the curvature does not help or reducing it to avoid the S-L error would compromise the integrity of the data, select as the reference path the table that allows the simulation to run; e.g., if the simulation runs with the **Lower Table** as the reference path but fails when the **Upper Table** is selected, use the **Lower Table**.

VS Visualizer: Slow to Load and Run with Long Roads and Thousands of Road: Animator Repeated Objects

Some end-users working with very long roads (i.e., hundreds or thousands of meters) have tried to link **Road: Animator Repeated Objects** such as guardrails and powerline poles for the full length of their road data and noted that VS Visualizer is slow to both load and run the animation. For end-users who have identified a need for this level of detail and are seeking improved animation performance, the recommendations are as follows:

- Shorten the length of the road.
- Reduce / reconsider the number of repeated objects.
- For continuous groups of objects such as guardrails, trees, poles, etc., consider the creation of new shape files that represent longer sections of these animation assets. For example, with a section of straight road that is 500 meters long, a single animation asset of a guardrail that is 100 meters long would only need to be used five times and therefore would consume fewer resources than 500 instances of a guardrail asset that is 1 meter long.

- For certain levels of detail, the best solution might be a custom scene in which all the details are built into a single `.obj` or `.osg` file. Mechanical Simulation ships several examples that use this technique.

It is important to remember that VS Visualizer assembles all linked animation assets and renders them all at once into a single scene; it does not load portions of the rendered data based on where the vehicle is, nor does it perform one or more preprocessing steps to check for the repeated use of a given animation assets and try to optimize the rendering of it.