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


# Running Multiple VS Vehicles in Simulink

This document describes how multiple vehicles from a VehicleSim (VS) product (BikeSim, CarSim, and TruckSim) can interact dynamically in a single Simulink model. The example simulation in this memo is based on datasets included in CarSim 8; similar examples also exist in TruckSim and BikeSim.

## Example Simulink Model with Two Vehicles

All VS products come with an example Simulink model that includes two vehicles that interact. (See the examples in the **Datasets** menu, category **Simulink** or category **Traffic and Sensors**). For example, CarSim 8 includes a Simulink model showing two vehicles interacting, with the second vehicle using adaptive cruise control based on the distance between the vehicles. Figure 1 shows a view from the animation.

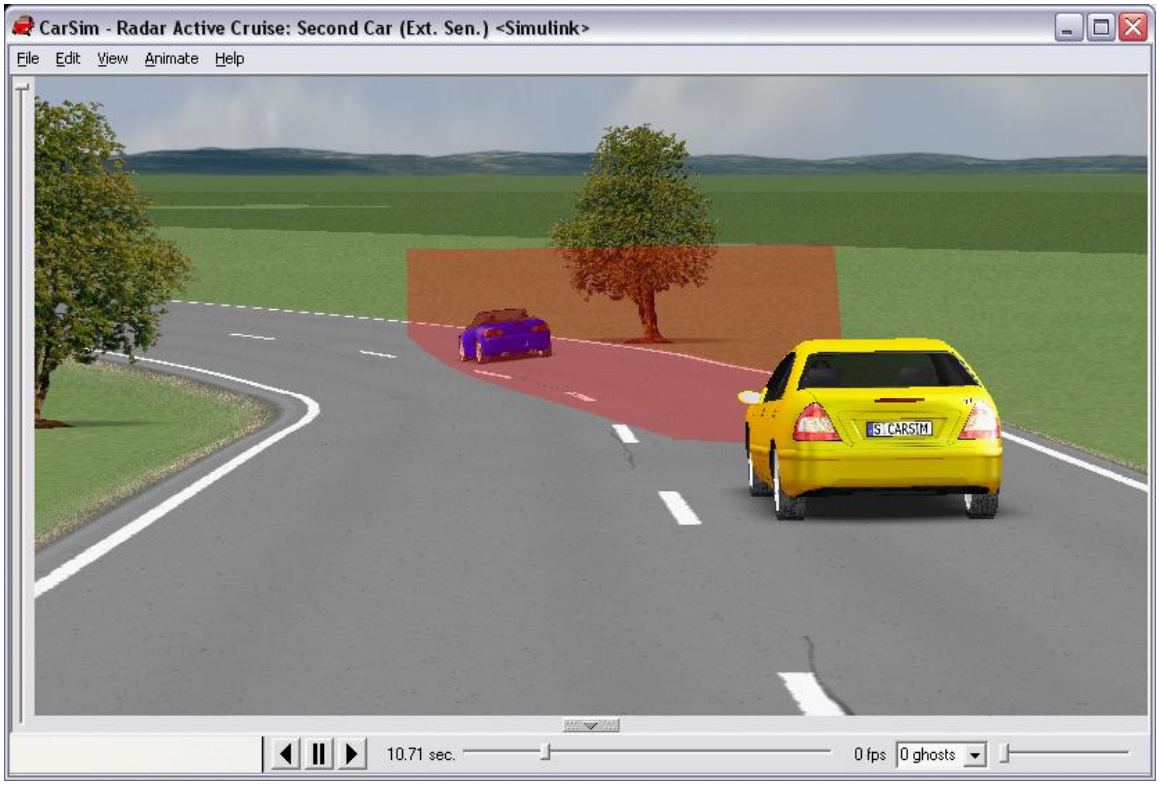


*Figure 1. Animation of two-vehicle simulation example in CarSim.*

Any simulation that involves multiple vehicles must have at least one **Run Control** dataset per vehicle that provides a description of that vehicle. For the CarSim example, the two vehicles are defined with the datasets:

- **{Simulink} Radar Active Cruise: First Car (Ext. Sen)**
- **{Simulink} Radar Active Cruise: Second Car (Ext. Sen)**

Before making a new run in Simulink involving multiple vehicles, there is an extra step necessary after installing the software. Navigate to each run that is referenced in the Simulink model (the two datasets noted above), unlock the screen, and click the **Send to Simulink** button. After this is done for both datasets, the Simulink model is ready to run with the two vehicles.

> **Note** The run can be initiated from Simulink, or from either **Run Control** dataset using the **Run Now** button.

## Background: Running One Vehicle Model in Simulink

Before going into the details of the two-vehicle example, it is helpful to understand how a VS browser such as `carsim.exe` works with Simulink for a single vehicle. For example, consider the Simulink ABS model included in CarSim that is shown in Figure 2. (Similar ABS models are included in BikeSim and TruckSim.)

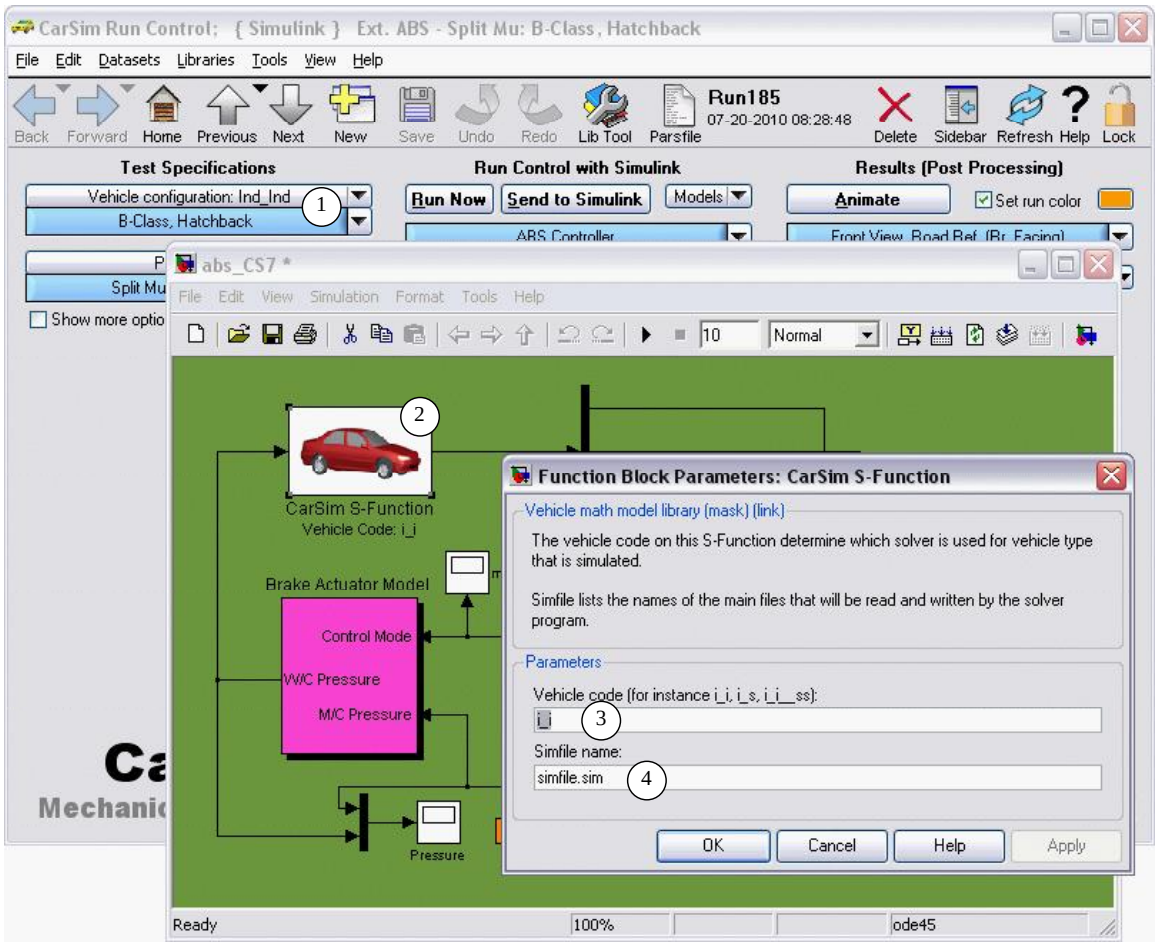


*Figure 2. Simulink model with one VS vehicle model.*

VS products such as CarSim include a group of dynamically linked library (DLL) solver program files, each optimized for a specific multibody vehicle configuration involving suspension types, number of axles, number of trailers, and other factors. The multibody configuration is identified by a *vehicle code*, associated with the vehicle dataset. In Figure 2, the vehicle dataset link shows that the vehicle code for the selected vehicle is `Ind_Ind` ①. (This code represents a 2-axle vehicle with independent suspensions front and rear.) A shorter code `i_i` was sent automatically to the Simulink model by the CarSim browser; this code is shown under the CarSim S-Function block ② and in the **Function Block Parameters** window ③.

The VS S-function block obtains information about the vehicle model from a simulation control file (simfile). The **Function Block Parameters** window shows the name of the simfile ④. (The name `simfile.sim` is set automatically as a default.)

Listing 1 shows the contents of the simfile for the example run from Figure 2.

*Listing 1. Example simfile for a single vehicle simulation.*

```
1. SIMFILE
2. INPUT D:\Product_Working\July_20\CarSim_Data\Runs\Run185_all.par
3. ECHO D:\Product_Working\July_20\CarSim_Data\Runs\Run185_echo.par
4. FINAL D:\Product_Working\July_20\CarSim_Data\Runs\Run185_end.par
5. LOGFILE D:\Product_Working\July_20\CarSim_Data\Runs\Run185_log.txt
6. ERDFILE D:\Product_Working\July_20\CarSim_Data\Runs\Run185.erd
7. DLLFILE C:\Program Files\CarSim803_Prog\Programs\Solvers\Default\i_i.dll
8. END
```

When Simulink opens the Simulink model, several things happen automatically:

1. Simulink queries the VS S-Function block (e.g., the CarSim block) to determine the number of input and output variables and other properties of the block.

    a. In response to the query from Simulink, the VS S-function reads the simfile using the specified name ④ to find the global pathname for a DLL file (`i_i.dll`, shown in line 7, Listing 1).

    b. The S-Function loads the DLL file identified in step 1a, and uses a function in the DLL to read the same simfile to obtain the name of an input parsfile (line 2).

    c. The DLL reads the parsfile identified in step 1b, obtaining the number of import and export variables that were specified from CarSim. The DLL also reads all information that will be used to characterize the vehicle and simulated test in the run.

2. Simulink checks for compatibility with the input and output controls that are set in the Simulink model. (Simulink generates an error message if they don't match.)

3. Simulink sends a terminate command to the S-Function block; the S-function responds by unloading the VS DLL.

Whenever a run is initiated, either from Simulink or the VS browser, the S-Function loads the DLL file again, repeating step 1 above. Thus, the simfile is always read at least twice for a run:

once to determine the number of import and export variables when Simulink loads Simulink model, and a second time when the run starts.

It is common to change vehicle properties or run conditions from the VS browser after a Simulink model is activated. If this is done, the **Send to Simulink** button must be clicked to make the browser rewrite the data files sent to Simulink.

# Considerations with Multiple Vehicle Models

The VS browser is designed to automatically load the appropriate VS solver from the family of DLL files. This applies whether the simulation is run with or without external software such as Simulink. Whether a vehicle model is running by itself or with others, the same process is followed for a specific vehicle model. The main consideration for handling multiple vehicle models is that each vehicle must be defined independently:

1. There must be an independent solver DLL for each vehicle.
2. Each vehicle must be set up with an independent **Run Control** simulation parsfile.
3. Each simulation parsfile (e.g., `Run101_all.par`) must be listed in an independent simfile.

## Multiple VS Solvers (DLL Files)

A VS solver DLL is intended to represent a single vehicle; it cannot represent two independent vehicles at the same time under Windows OS. Each vehicle will automatically use a different DLL if the vehicles have different configurations. For example, if one vehicle has two independent suspensions (code = `i_i`) and the other has an independent front and solid-axle rear (code = `i_s`), then two different DLL files are automatically selected for the two configurations.

When there is a need to simulate two vehicles at the same time that have the same VS configuration (e.g., `i_i`), then a simple method for defining a new DLL is to make a copy with a different name (e.g., `i_i2.dll`) and use the copy for one of the vehicles. Even though two DLL files are identical, the Windows OS will handle them as being completely independent and allow them to maintain independent sets of vehicle parameters and variables.

## Multiple Simfiles and Parsfiles

In normal operation with a single vehicle, the VS solver writes a new simfile named `simfile.sim` that in turn references a parsfile generated from the **Run Control** dataset currently in view. (The simfile in written in the root folder of the VS database currently in use, which is also the current working directory.)

If two separate vehicles are to be simulated simultaneously, then each needs a separate simfile and parsfile. A simple way to set up multiple vehicle models that can run at the same time is to create a separate **Run Control** dataset to represent each vehicle, as described for the CarSim example below. Each dataset will produce a uniquely named parsfile with all of the information needed by the DLL to simulate the vehicle, the controls, and the environment.

When running multiple vehicles, it is a good idea to specify a uniquely named simfile in order to avoid conflicts with simfiles generated for other runs. In this example, the simfile names are similar to the Simulink file name with `car_1` or `car_2` added at the end.

> **Note** The contents of the simfile and data parsfiles are described in more detail in the *VS Solver Programs Reference Manual.*

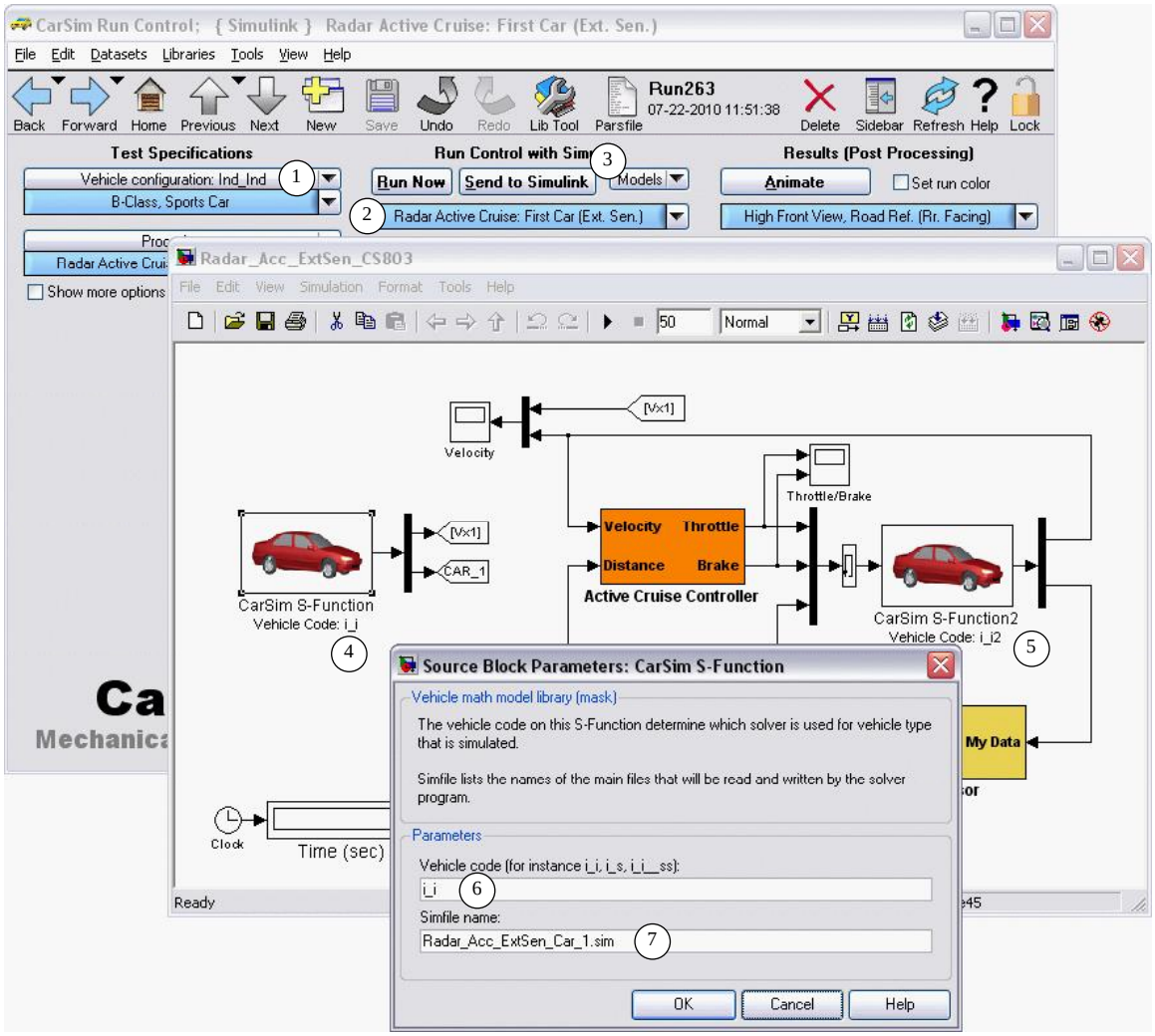


*Figure 3. Example Simulink model with two vehicles that interact.*

## Example in CarSim: the First Vehicle

Figure 3 shows one of two **Run Control** datasets that connect to the example Simulink model. Clicking on the **Send to Simulink** button ③ brings up a Simulink model that has two CarSim S-Function blocks. Double-click on the block for the first car ④ to see the associated parameters in a **Source Block Parameters** window. The two parameters for the block are the vehicle code ⑥ (`i_i`)and the simfile name ⑦ (`Radar_Acc_ExtSen_Car_1.sim`). (The simfile name is based on the Simulink model name for the radar-based adaptive cruise control with an externally

defined sensor, `Radar_Acc_ExtSen_CS803`, and `Car_1`. The name is made similar to the Simulink MDL file to help identify it if any debugging is required.)

Clicking on the link to the Model dataset ③ shows the **Models: Simulink** dataset used to identify the Simulink model from the point of view of the first car (Figure 4).

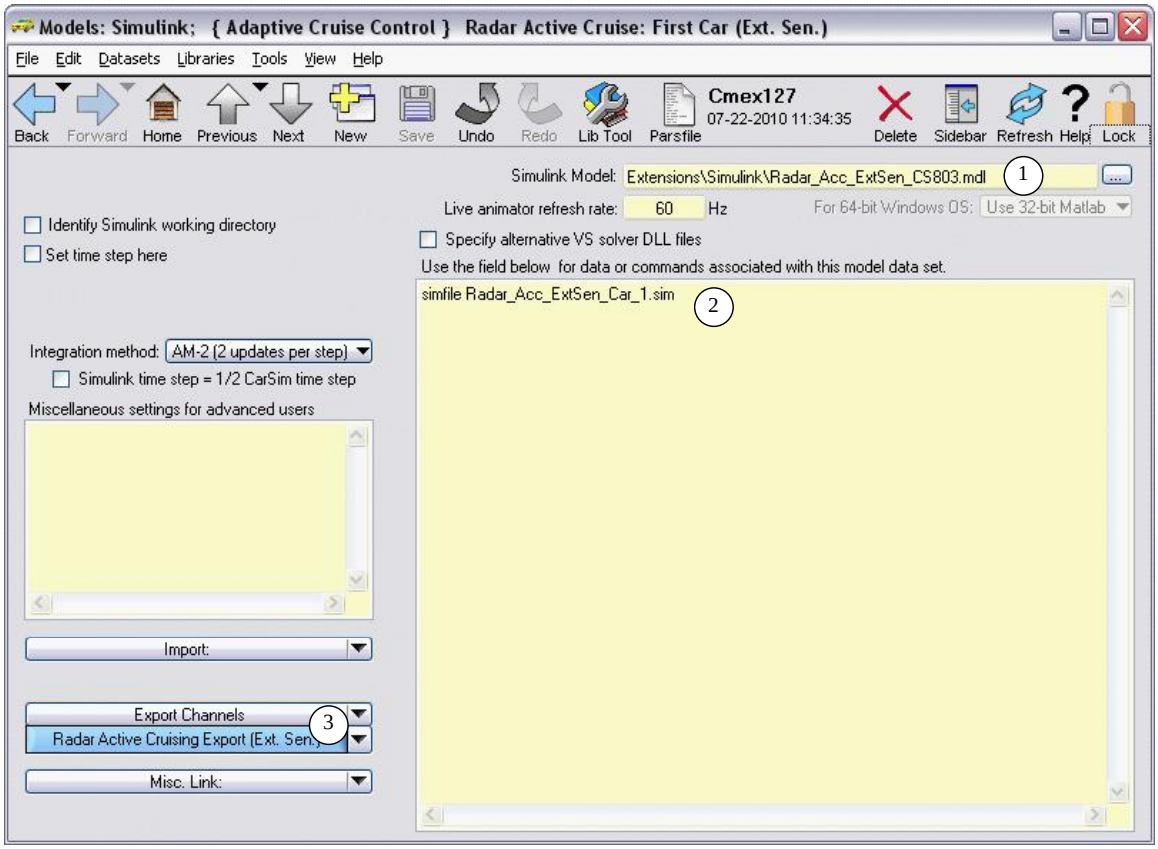


*Figure 4. Models: Simulink screen for the first vehicle.*

This dataset has just three pieces of information. First, the name of the Simulink model file is specified ①. The lead vehicle will not receive any information from Simulink, but will provide its location via export variables ③ to support a controller based on the distance between the two vehicles.

Because the Simulink model has two CarSim vehicles, this dataset will also specify the name of the simfile with all information about the first car. The miscellaneous field ② has line with the keyword `simfile` used to specify the name `Radar_Acc_ExtSen_Car_1.sim`. Note that the simfile name here ② matches the name specified in the Simulink model (⑦ in Figure 3).

## Example in CarSim: the Second Vehicle

A different **Models: Simulink** dataset links to the same the Simulink model from the point of view of the second vehicle (Figure 5). This dataset specifies the same Simulink file ① as the dataset for the first vehicle (Figure 4), but has different settings for the vehicle.

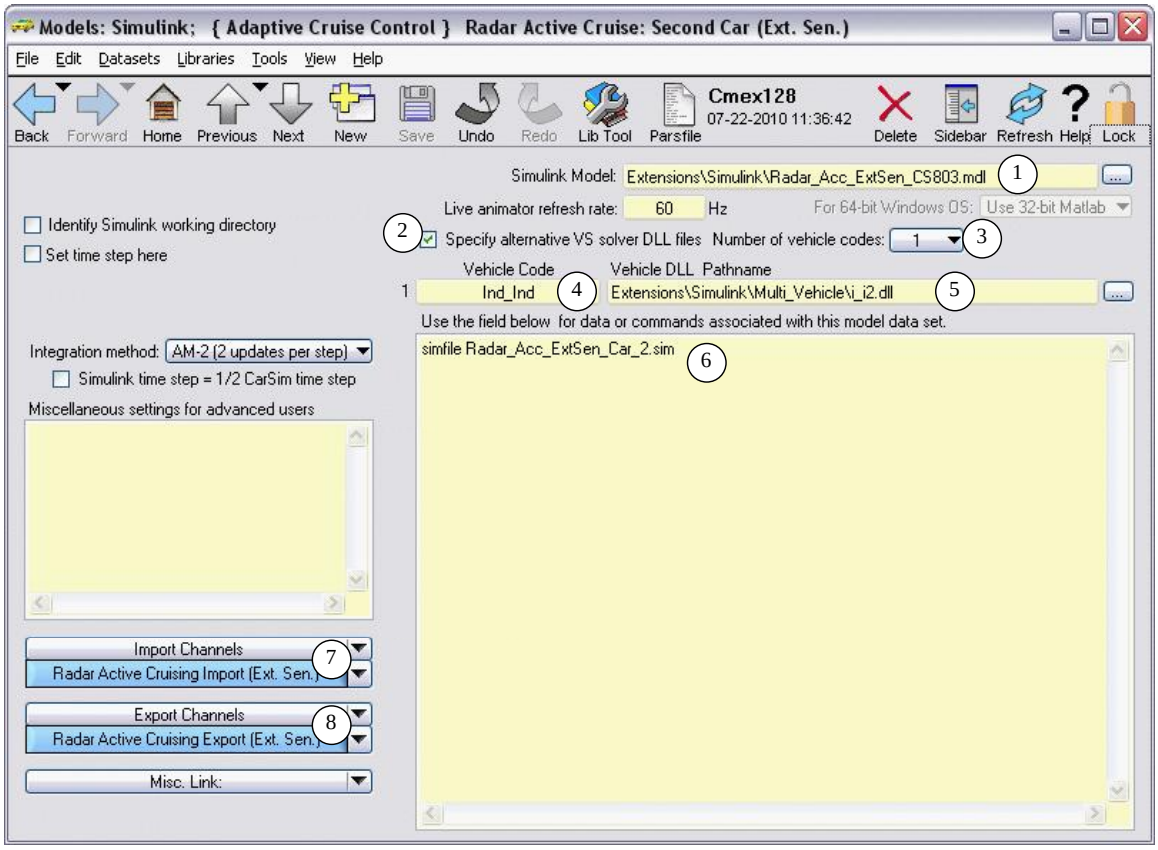


*Figure 5. Models: Simulink screen for second vehicle.*

As with the first vehicle, a custom simfile name is specified: `Radar_Acc_ExtSen_Car_2.sim` (6). This name matches the name shown in the **Function Block Parameters** window for the second vehicle block in the Simulink model (Figure 6).

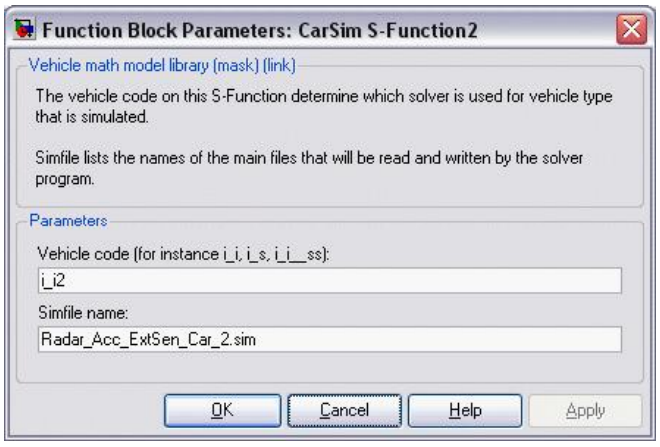


*Figure 6. Function block parameters for second vehicle.*

Both vehicles in the simulation have the same configuration: independent front and rear suspensions. In order to allow two independent vehicles to run simultaneously, a copy of the `i_i.dll` file was made, with the pathname shown (5) (Figure 5).

**Note** The solver DLL files are located in the PROG folder for the software; e.g., for CarSim 8.03 they are in the folder `CarSim803_Prog\Programs\Solvers\Default`. (The PROG folder is typically installed in the Windows **Program Files** folder.)

To use the alternate solver file `i_i2.dll`, a box is checked (2). When checked, a pull-down control is available for setting the number of vehicle types that will be associated with alternative solver files (3). In this case, the control is set to 1, causing data fields to be shown for one vehicle code (4) and one pathname (5). The settings in these fields indicate that when a vehicle is selected on the **Run Control** screen with code `Ind_Ind` (4), then the associated DLL file is to be `Extensions\Simulink\Multi_Vehicles\i_i2.dll` (3).

The file name `i_i2.dll` from this screen is also set in the **Function Block Parameters** field (Figure 6).

In the simulation, the second vehicle has a driver assistance controller that uses the locations of the two vehicles; to support this, there are links to datasets that specify variables for import (7) and export (8) relative to the Simulink model.

## Summary of Method for Simulating Multiple Vehicles

The general method for building a Simulink model that involves two or more vehicles can now be summarized.

1. Each vehicle model in the Simulink model (an MDL file) must be represented by a copy of the vehicle S-Function block provided with the VS package. (E.g., the CarSim S-Function for CarSim). For example, in Figure 3, there are two copies of this block.
2. Each vehicle must be defined in the VS database with a dataset in the **Models: Simulink** library with a link to the Simulink MDL file, and links to sets of import and output channels that match the control inputs and outputs for the corresponding block in the Simulink MDL model.
3. Each S-Function vehicle block should be associated with a different VS DLL file. If multiple vehicles have the same configurations (e.g., `i_i`), then copies must be made of the DLL (e.g., `i_i2.dll`) and links should be made to the copies, as done in the example for the second vehicle in the simulation.
4. Each **Models: Simulink** dataset should specify a uniquely named simfile, using the `simfile` keyword to specify the name as shown in Figure 5 and Figure 4.
5. The S-Function block parameters from the Simulink MDL file should be edited manually to match the DLL name and simfile name specified in the corresponding **Models: Simulink** datasets.
6. A dataset should be made for each vehicle from the **Run Control** screen. As a minimum, each **Run Control** dataset should include a link to the vehicle dataset for one of the S-function blocks in the model, and a link to a matching **Models: Simulink** dataset for the same block.

7. The **Send to Simulink** button should be clicked from each **Run Control** dataset used in the Simulink model. Any errors reported by Simulink must be fixed before any runs can be made.

After these steps have been completed, the run can be initiated from any of the contributing **Run Control** datasets.