CarSim – Simulink Model with Two Vehicles

This simulation is based on datasets included in CarSim; it uses two **Run Control** datasets, each linked to a different vehicle, and both linked to the same Simulink model. The two **Run Control** datasets are in the **Datasets** category **Simulink and LabVIEW Models** with titles that begin with **Radar Active Cruise** (1) (Figure 1).

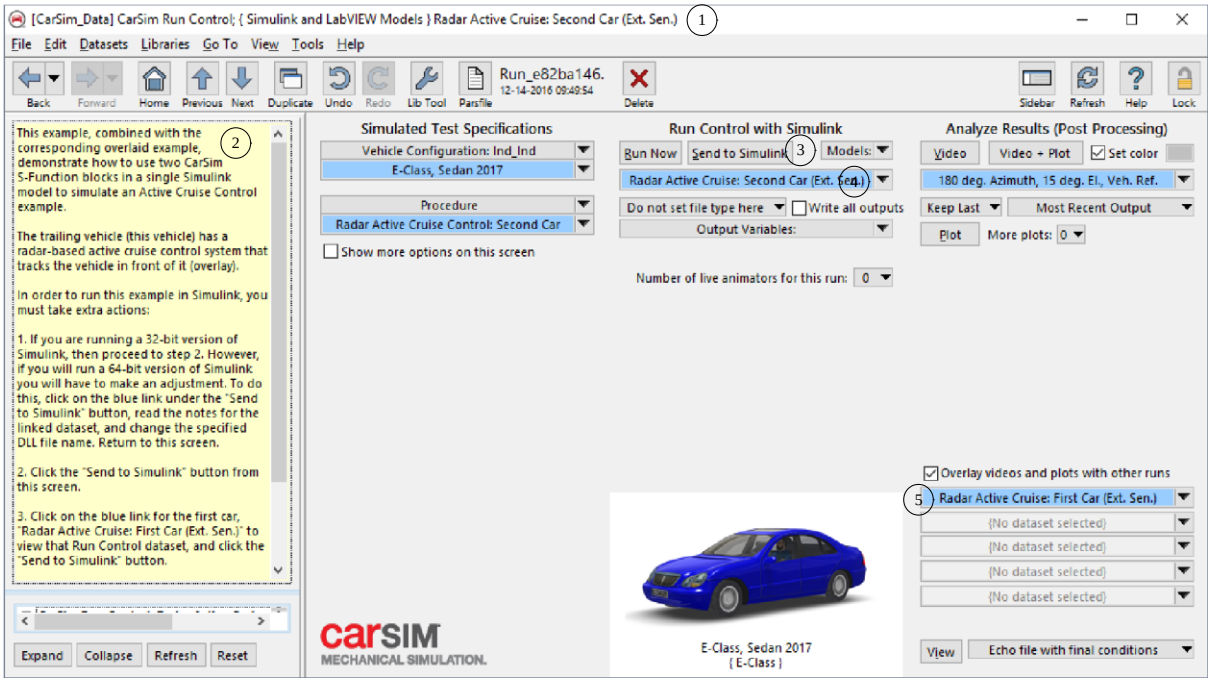

Figure 1. Run Control dataset for second car in multi-vehicle example.

Both of the **Run Control** datasets have extensive notes (2) with the steps needed to make a new run from Simulink. Each has a link to the other (5) for the purpose of showing both vehicles in the same video. Each is linked to a **Models: Simulink** dataset that identifies the Simulink model (4). Although both **Models: Simulink** datasets specify the same Simulink model, they are different in how they connect vehicle data to the model, and how they maintain two independent vehicle math models. View the video for the simulation by clicking the **Video** button from either of the two **Run Control** datasets (Figure 2).

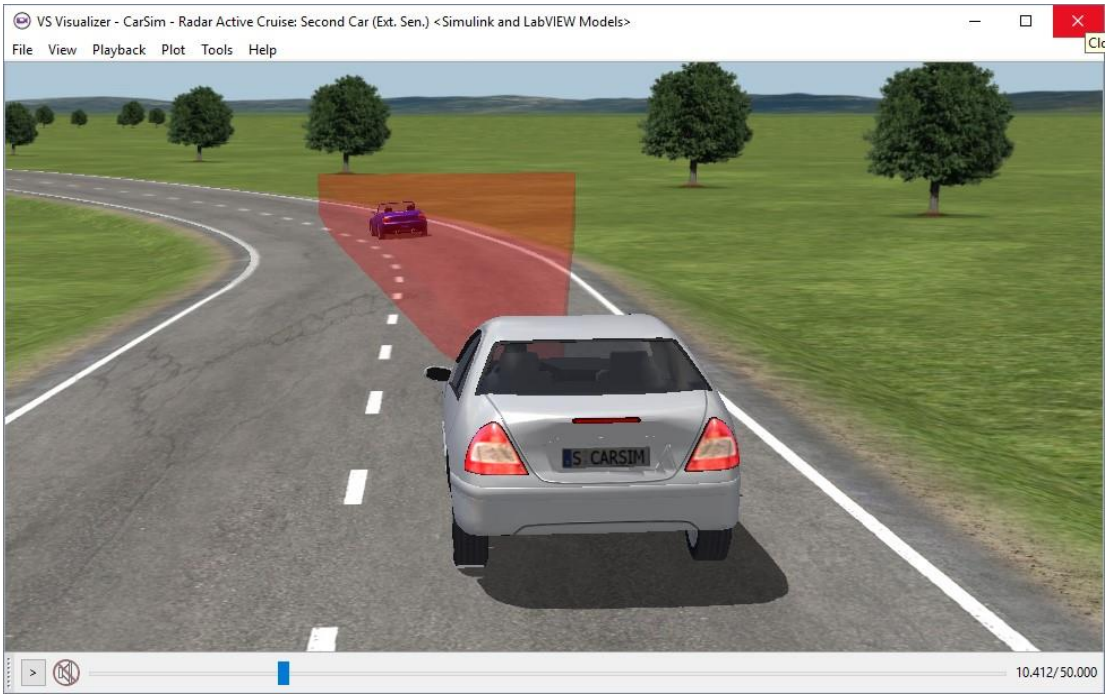

Figure 2. video with the camera following the second car.

Before making a new run in Simulink involving multiple vehicles, there is at least one extra step necessary after installing the software. While viewing one of the unlocked datasets (e.g., Figure 1), click the **Send to Simulink** button (3). Next, click the blue link for the second dataset (5), and click the **Send to Simulink** button for that dataset as well (it must also be unlocked). The Simulink model (Figure 3) is now ready to run with the two vehicles if you are using a compatible version of MATLAB/Simulink (at the time this memo was last revised, the example was prepared for 32-bit versions).

Notice that there are two vehicle blocks in the model. Double-click on the second one (2) to view the parameters for the function block (Figure 4). In most of the Simulink models using a VS S-Function, these parameters are set automatically. However, in this case, they were originally set by hand. The vehicle code is set to i_i2. The S-Function will use this to find a VS Solver file with the name i_i2.dll.

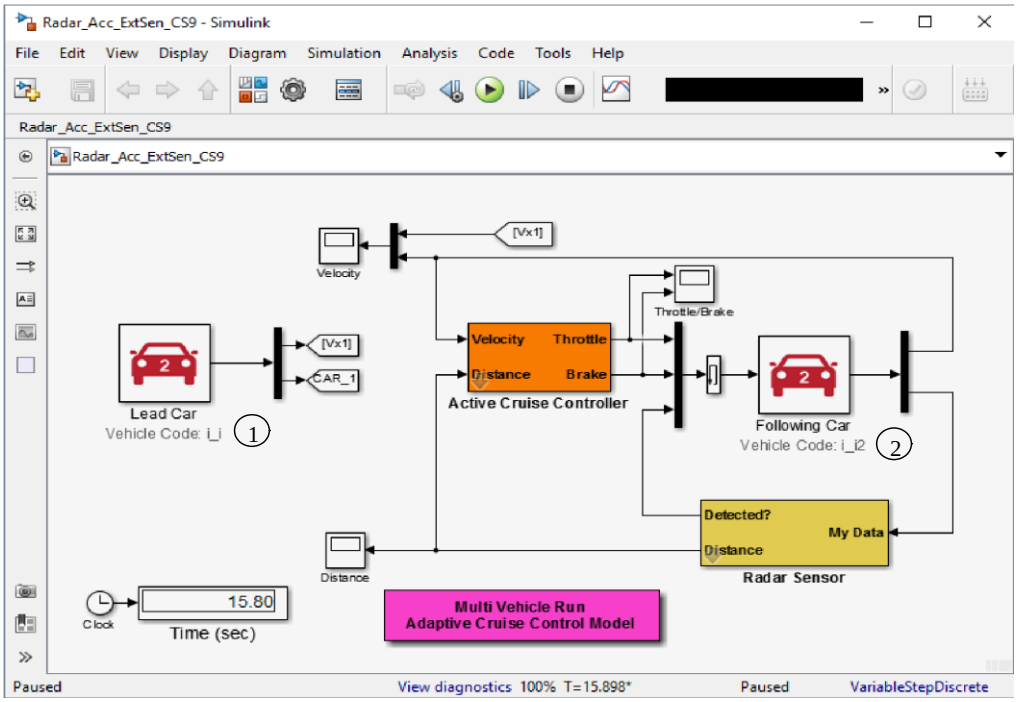

Figure 3. CarSim – Simulink model with two VS S-Functions.

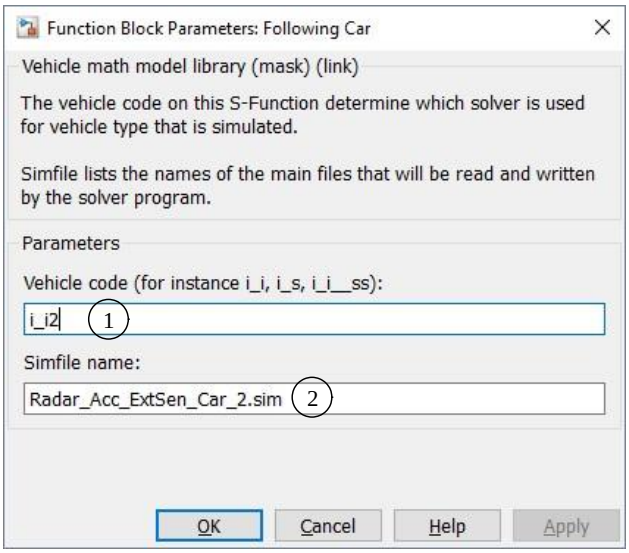

Figure 4. Parameters for the S-Function block for the second vehicle.

The file i_i2.dll is not a standard DLL in CarSim, so it must be specified from the CarSim database. Go to the **Run Control** dataset for the second vehicle (Figure 1) and click on the blue link under the **Send to Simulink** button to view the **Models: Simulink** dataset (Figure 5).

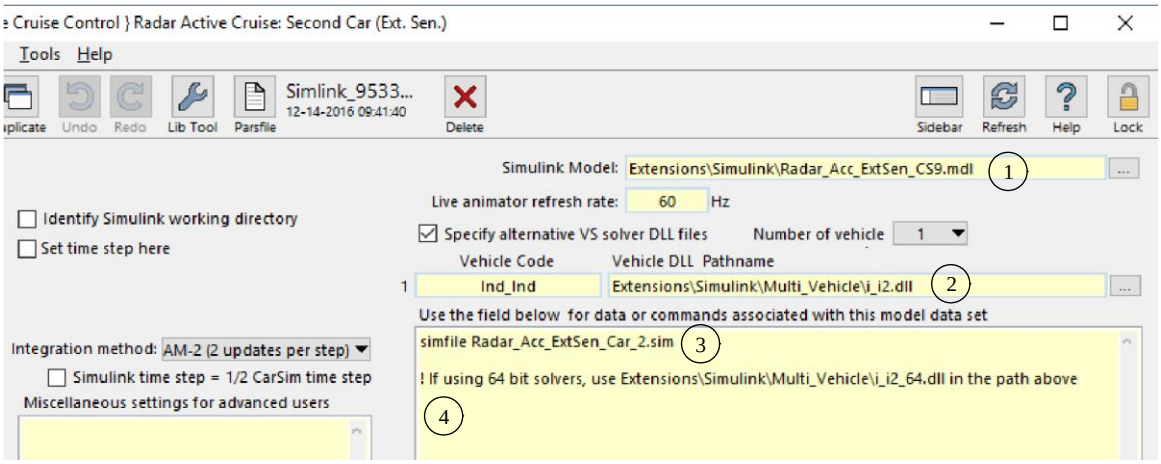

Figure 5. Models: Simulink dataset that identifies the DLL to be used for the second vehicle.

This dataset identifies the Simulink model file (1), the specific DLL file i_i2.dll to use for the vehicle type ind_ind (2), and the name to use for the Simfile (3). (A Simfile is used for every new simulation. It specifies the model type, names of i/o files and other information.) The Simfile name used in this dataset (3) must match the name specified in the Function Block Parameters window (2) (Figure 4). It is essential that the DLL file be compatible with Simulink in terms of 32/64 bit. If it is not, Simulink will crash when you try to run the simulation. A note in the miscellaneous yellow field identifies the DLL file needed for simulation with 64-bit Simulink. In this simulation, change i_i2 to i_i2_64 if you will run with 64-bit Simulink.

After you have clicked the **Send to Simulink** button for the two **Run Control** datasets, select the **About**… option from the **Help** menu and click on the underlined database name (Figure 6). This shows the database folder in Windows (Figure 7). Notice that the two custom Simfiles ((1) and (2)) were written in this folder.

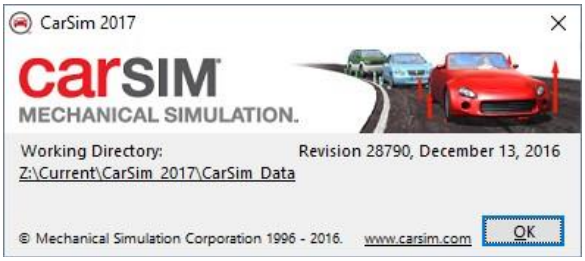

Figure 6. Click on the underlined pathname to view the folder in Windows.

Figure 8 shows the **Models: Simulink** dataset for the lead vehicle. In comparing this with the dataset in Figure 5, you can see it is a little simpler. It identifies the same Simulink model file (1) and another custom Simfile name (2), which was also written into the database folder (Figure 7). However, the **Models: Simulink** dataset for the lead vehicle does not specify a custom DLL pathname; it relies on the automatic built-in behavior of the VS Browser, which gathers information about the vehicle configuration from the vehicle assembly dataset linked to the run.

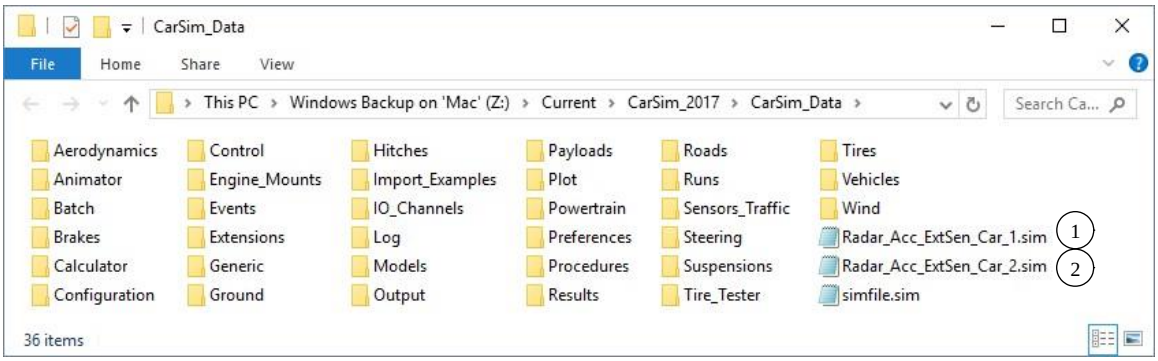

Figure 7. Simfiles are written using the specified names for the example.

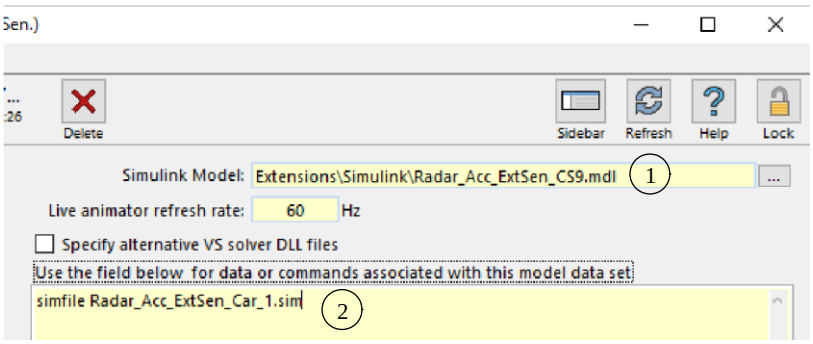

Figure 8. Models: Simulink dataset for the first vehicle

Considerations with Multiple Vehicle Models

The VS Browser is designed to automatically specify the appropriate VS Solver from the family of vehicle math model 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 a unique VS Solver DLL for each vehicle used in the simulation.
2. Each vehicle must be linked to an independent **Run Control** dataset.
3. Each **Run Control** dataset must be assigned to use a unique Simfile.

Here is a summary of the general method for building a Simulink model that involves two or more vehicles:

1. Create a vehicle dataset to be used in a co-simulation environment with the ~~VS product~~ (~~e.g.,~~ CarSim) and Simulink, and link that vehicle dataset to a **Run Control** dataset.
2. For each of these Run Control / vehicle dataset combinations, a corresponding dataset must be created in the **Models: Simulink** library and linked to the **Run Control** screen.

The **Models: Simulink** dataset contains a link to the Simulink file (MDL or SLX) that will include the multiple S-Function blocks. The **Models: Simulink** dataset also has links to sets of import and output channels (i.e., the I/O Array) that match the control inputs and outputs for the corresponding VS S-Function block in the Simulink model.

3. Each VS vehicle model running in the Simulink model must be represented by a copy of the vehicle S-Function block provided with the CarSim S-Function.
4. Each S-Function vehicle block should be associated with a different VS DLL file. If multiple vehicles have the same configuration (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. The copy must be made of a DLL that is compatible with the Simulink version (32-bit or 64-bit).
5. 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 8.
6. The S-Function block parameters from the Simulink model should be edited manually to match the DLL name and Simfile name specified in the corresponding **Models: Simulink** datasets.
7. The **Send to Simulink** button should be clicked from each **Run Control** dataset used with 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.