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

External Models and RT Systems

CarSim, TruckSim, and BikeSim support many options for extending models using external software, beyond the tools that are built-in to the Windows programs and database. Options to use other software are specified using datasets from **Models** libraries that are a part of the VS Browser.

Note *VS Browser* is the generic name for the main program in CarSim, BikeSim, or TruckSim (e.g., `carsim.exe`).

This document describes all the **Models** screens and the way they interact with the **Run Control** screen of a VS Browser. CarSim, TruckSim, and BikeSim have 11 **Models** libraries. **Models** library screens provide up to four kinds of information:

1. Settings for the numerical integration used in the VS Math Model when working with the external software.
2. Lists of import and export variables that are used to connect with external software.
3. Locations of configuration and project files used by external software.
4. Communication settings used to send files between a Windows host computer (running the VS Browser) and another target computer, typically running a real-time (RT) operating system.

This document is organized to describe features common for all **Models** screens first, followed by documentation for screens that are specific for external tools. Here is the general organization:

- The first three sections describe the way **Models** datasets interact with the **Run Control** screen and features common to most **Models** screens.
- The next four sections describe the Simulink and LabVIEW screens, screens for custom extensions made using Windows, and the FMU/FMI screen.
- The next five sections describe screens used for RT systems. Most of the controls on these screens have already been covered, so the sections focus on features that are specific to a relevant RT system.

The Run Control Screen

The **Run Control** screen has one or more buttons for controlling simulation runs. A drop-down list is available for specifying an optional model library, as shown in Figure 1. When a link is made to a **Models** library, the appearance and behavior of the buttons in this section of the **Run Control** screen are usually altered.

Run Buttons

When there is no linked library, the VS Browser displays the single button shown in Figure 1: **Run Math Model**. When you click this button, the browser loads the VS Solver library and builds a VS Math Model that runs a simulation.

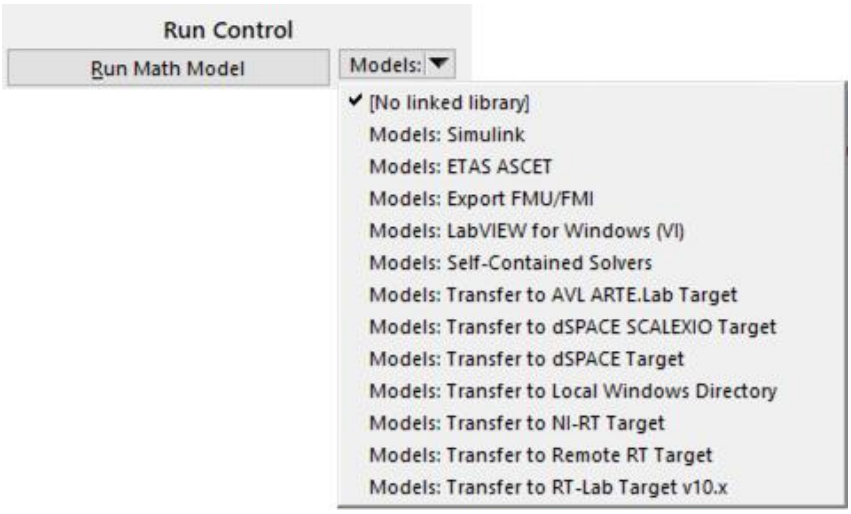


Figure 1. Linking to a Model library from the CarSim Run Control screen.

If a link is made to one of the available **Models** libraries, then the appearance of the **Run Control** screen is adjusted to support the type of library that is selected. Other buttons perform functions such as transferring data to another folder or computer, receiving data, opening a model in external software (e.g., Simulink), or making a run using external software and possibly a second computer. Also, a blue link to the selected dataset from the linked library is shown under the buttons.

Buttons that are shown when a link is made to a dataset from specific **Models** libraries are described in the sections covering the **Models** library of interest.

Live Video (Animation)

If the linked **Models** library supports Simulink or real-time (RT) simulation (with an RT computer or a driving simulator (DS) option), then a drop-down control is available on the **Run Control** screen for selecting the number of live animators. If the selected number is one, then two more controls appear for controlling the video (Figure 2). These controls are described in the documentation of the **Run Control** screen.

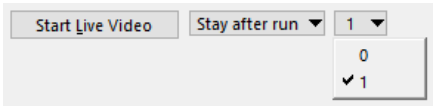


Figure 2. Controls when there is one live animator.

Controls On Nearly All Models Screens

Figure 3 shows the **Models: Simulink** screen with a typical setup. Nearly all of the numbered items on the screen appear on every other **Models** screen and are described below in detail.

Integration Method and Time Step

The VS Solver calculates variables as the run proceeds by using numerical integration (see *VS Math Models Reference Manual* or the technical memo *Numerical Integration in VS Math Models* for

details). The numerical integration is performed by increasing time T in the model by time-step increments.

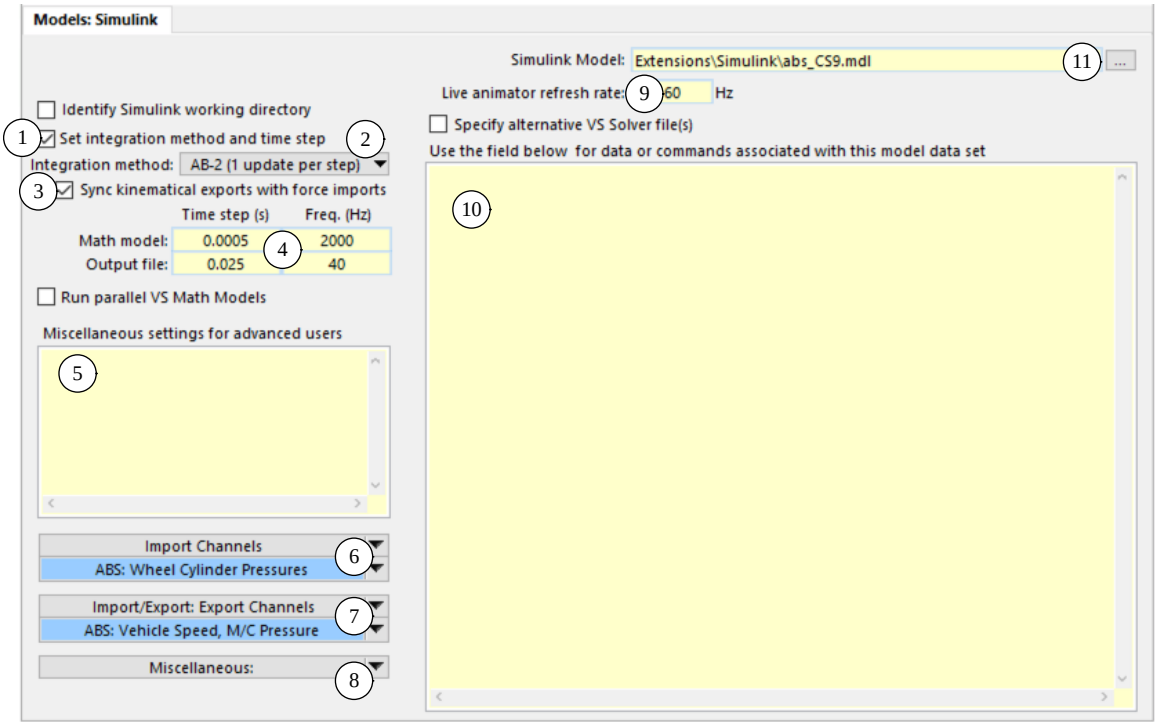


Figure 3. Models: Simulink screen with basic controls shown.

The numerical integration method and time step parameters can be specified in three places in the database:

1. The default numerical integration method and time step are set in the **Preferences** library (to view the current **Preferences** dataset, select **Preferences** from the **Tools** menu). The settings from the most recently viewed **Preferences** dataset will be used if no other values are specified in the datasets used for a simulation.
2. The numerical integration method and time step values can be set on all **Models** screens (check the box (1) to see the controls). This is recommended when using any external model; in many cases the external software requires a specific integration method and time step that are not necessarily standard for other simulations made from the same database. The time steps specified in a linked **Models** dataset will override the values from the most recently viewed **Preferences** dataset.
3. Time step values can be set on the **Run Control** screen. The time steps specified on that screen will override any other values.

Outputs can be written at an integer multiple of the model time step to reduce the size of the output files. The output variables are written into a binary file that is associated with a VS or ERD text file that has header information needed to read the binary file.

The following controls ((1) - (4)) are available on every **Models** screen.

① **Set integration method and time step**. Check this box to view controls involving the numerical integration method and time steps (② - ④).

As noted above, if the box is not checked, the integration method and time step from the most recently view **Preferences** dataset will likely be used in all simulations involving this dataset. When you click the box to change it from unchecked to checked, the VS Browser automatically sets the method and time steps that are displayed (controls ② - ④) to match the specifications from the most recently view **Preferences** dataset.

Note	A handy trick to see the default method and time step from **Preferences** is to uncheck this box, refresh the screen (F5), then check it. The values shown for the controls ② - ④ are what will be used if the box is not checked. After seeing the information, use Undo (Ctrl+Z) as needed to restore the dataset to its original state.

② Drop-down list for selecting a numerical integration method (keyword = `OPT_INT_METHOD`). VS Math Models support six different methods for numerically integrating differential equations (Figure 4).

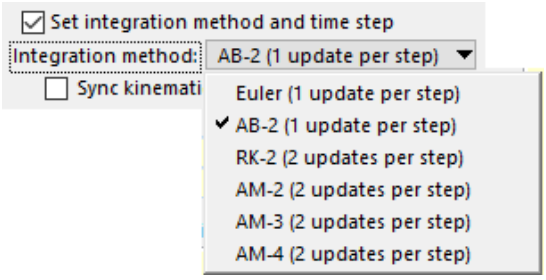


Figure 4. Drop-down control to set numerical integration method.

The Euler and AB-2 (Adams-Bashforth) methods calculate all model variables once per time step. A typical time step with either Euler or AB-2 is 0.0005s. The Euler method is the least accurate, but is sometimes useful for diagnosing problems.

The AB-2 method is a second-order method that is more accurate than Euler, and is well suited for exchanging data with external models that work better with rapid communication. This method is also good for simulations that use VS Commands to extend the model. Overall, the AB-2 is recommended as the best general-purpose method, with a time step of 0.0005s.

The other four methods calculate the model variables twice per time step: once at the major step, and a second time at the half step. For example, if the time step is 0.001s, the calculations are made at intervals of 0.0005s. Of these, the recommended method is AM-2 (Adams-Moulton second-order method). AM-3 and AM-4 are third- and fourth-order versions that might be better for some conditions that are largely continuous. RK-2 (Runge-Kutta second-order method) was the method available in early versions of the software, and it is sometimes better suited for simulations where a half-step method is desired but system will exhibit discontinuous behavior.

For more information about these methods, please see the technical memo *Numerical Integration in VS Math Models*, available from the **Help** menu.

(3) Checkbox to enable an option for the selected integration method. The option that can be enabled depends on the integration method.

Synchronize kinematical exports with force imports

In the case of Euler or AB-2 integration, a checkbox (keyword = `OPT_IO_SYNC_FM`) is used to choose between two methods for controlling the exchange of information with external software such as Simulink. This choice has no effect unless external software is used in which the VS Solver appears as a block (e.g., a Simulink S-Function, a functional mockup unit (FMU), etc.), and communication with the VS Solver block includes both import and export variables.

When unchecked (`OPT_IO_SYNC_FM` = 0), each time the VS Solver block is activated all calculations are made for time T, where T is provided by the external software. This sequence works perfectly if the variables imported into the VS Solver block are based on factors external to the vehicle model, such as clock time. However, if the variables being imported are calculated based on variables that are exported, then the imported variables have a time lag because the external model had to use export variables calculated at the previous time step. This type of connection occurs when parts of the multibody model are defined externally, such as tires, springs, powertrain, etc. Deflections and speeds exported from the VS model are used to calculate forces and moments that are provided as imports to the VS model at the next time step.

When the box is checked (`OPT_IO_SYNC_FM` = 1), an alternative sequence is used to calculate some of the variables that are exported, such that forces and moments calculated externally are sychronized with the rest of the model. In this case, the calculation sequence is modified to provide kinematical variables (variables with units of m, m/s, rad/ and rad/s) for the next time step, at `T + TSTEP`. Forces and moments calculated externally will be synchronized when imported at the next step.

For more information, please see *VS Math Models Reference Manual*.

External time step = ½ VehicleSim time step

In the case of the methods that calculate the model variables twice per time step (RK-2 and the AM methods), this checkbox determines the time interval used to communicate with external software such as Simulink (keyword = `OPT_IO_UPDATE`).

When checked, the communication time step is 1/2 of the step set for the numerical integration. When communicating with external software, you can either exchange information at the major time step (e.g., 0.001s) or the half-step (0.0005).

1. Communicating whenever possible means the external model is performing twice as many computations. For a complex extension, this can slow down the simulation time. Better computation speed can be obtained if you run the external software with the same time step as indicated on the **Run Control** screen. In this case, uncheck the box.

2. If the extension involves high frequencies, then any delay between the transfer of information between the VS Solver and the extension can introduce a numerical instability. Numerical instability might be avoided by updating the external equations at 1/2 the time step indicated on the **Run Control** screen.

> **Note** Rather than checking this box to communicate with external software at the half-step, Mechanical Simulation recommends using the AB-2 fixed-step method with half the time step (e.g., 0.0005s). This provides a more consistent interface to the external software.
>
> The `OPT_IO_UPDATE` option is a legacy feature that was needed in earlier versions of the software when RK-2 was the only available integration method.

See *VS Math Models Reference Manual* for more information about the numerical methods and the `OPT_IO_UPDATE` parameter.

(4) Control of time interval for math model and output file.

The time step can be specified using either seconds or frequency. The browser automatically maintains the correct inverse relationship between these two values. If you modify the time step, the frequency is calculated and updated; if you modify the frequency, the time step is calculated and updated.

> **Note** The time step is the value passed to the math model: keyword = `TSTEP`.

As with the model time step, the VS Browser automatically maintains the correct inverse relationship between time step and frequency for the output file. It also guarantees that these numbers are multiples of the internal time step used in the VS Math Model.

The output time step is a multiple of the internal time step; the multiple is the system parameter `IPRINT` in the VS math model. For example, if the output time step is 0.025s (keyword = `TSTEP_WRITE`) and the numerical integration time step is 0.0005s (keyword = `TSTEP`), then `IPRINT` is set automatically by the VS Browser to 50.

The VS Browser automatically enforces the restriction that `IPRINT` must be an integer. When you change the model time step, you will sometimes see that the output time step is refreshed as needed to maintain the integer relationship.

> **Alert** Most parameters in the VS Solver can be specified with numbers or formulas. However, the four fields that specify the time step for the simulation and output file are also used to automatically calculate numbers, as described above. Because of the automatic calculations, each of these fields must contain a numerical value when they are visible. Blank fields and formulas are not supported.

Other Controls on Nearly All Models Screens

(5) Miscellaneous field. This field has no specific purpose but can be used to globally specify parameters and options that should always be applied when a run is made using this dataset.

(6) Link to a dataset. Traditionally this is used for an **I/O Channels: Import** dataset that specifies the variables that an external model (e.g., a Simulink model) is expecting to provide to the VS Solver. It can also be used for an **I/O Channels: Ports** dataset if working with an external model that uses multiple ports.

(7) Link to a dataset. Traditionally this is used for an **I/O Channels: Output** dataset that specifies the variables that an external model (e.g., a Simulink model) is expecting to receive from the VS Solver. It can also be used for an **I/O Channels: Ports** dataset if working with an external model that uses multiple ports.

(8) Miscellaneous link to a dataset that should always be associated with this model extension.

(9) Refresh frequency for live animation. This value is used when the model extension involves real-time operations with live animation using VS Visualizer. This value is sent to VS Visualizer when it is started.

(10) Field for inserting optional VS commands (or other data). This field is provided as a convenient location for adding equations for existing Import variables, adding new variables, extending the models in other ways, all without using external software. The VS commands are described in *VS Commands Manual*.

(11) Pathname for an external model or program. This field (and adjacent file browser button) exists in most of the **Models** screens, with different interpretations. For the example shown in Figure 3, it identifies a Simulink model file. With other Model screen, it will identify a type of file associated with that screen.

Settings on Many Models Screens

Figure 5 shows the **Models: Simulink** screen with all of the visible controls numbered. The basic settings were described in the previous section ((1) - (11)); this section describes the additional settings ((12) - (16)). Although the example screen is for Simulink, most of these controls are available in other screens as well.

Working Directory

(12) Checkbox and data field for specifying a working directory. When running with other software (e.g., Simulink), the VS Browser can send data to an arbitrary location, as might be needed for complicated external software models. You can type the directory name in directly, or use the adjacent browser button to select the folder.

On this screen, the field is visible and active only if the checkbox is checked. However, on some other screens, the directory is required, and there will be no checkbox.

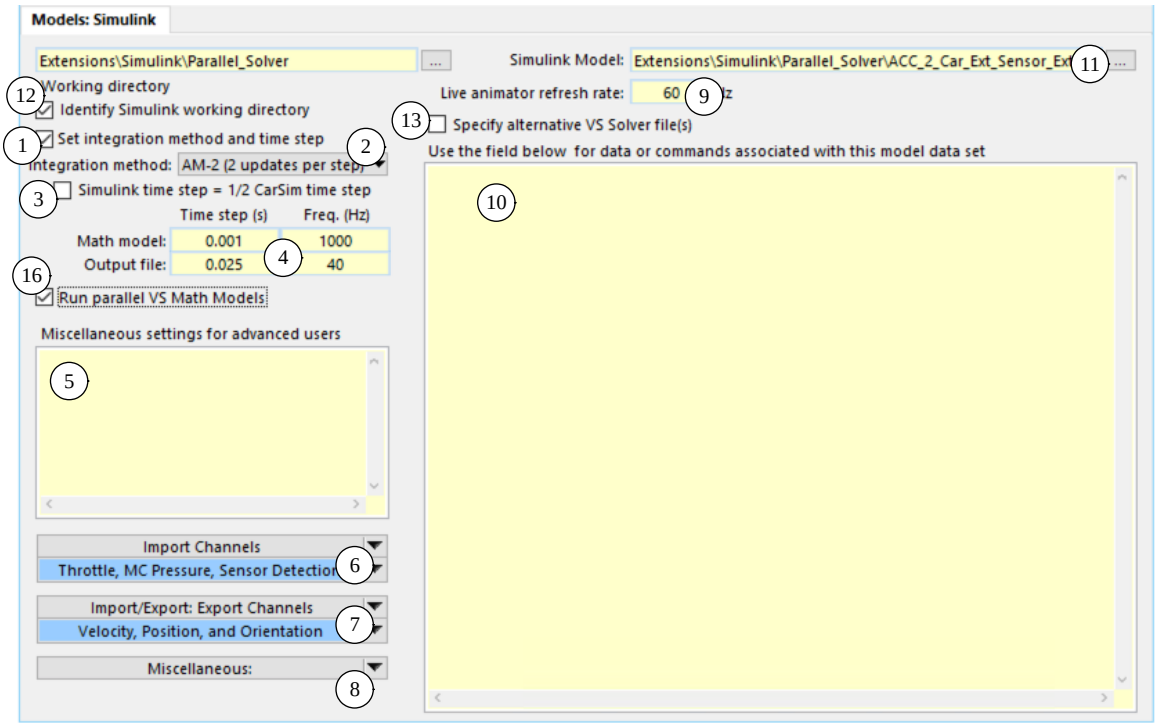


Figure 5. The Models: Simulink screen, including advanced controls.

> **Note** If the external model contains an initialization routine that changes the working directory (e.g. a `PreLoadFcn` entry in Simulink) to a different directory than is specified in the VS Browser, then the simfile may not be able to be found, and the simulation will not operate as intended. If this occurs, you either need to make sure that the working directory field in the VS Browser reflects the directory specified in the initialization routine, or you should modify the initialization routine to restore the original working directory when it concludes.

Alternative VS Solver Libraries

Some of the **Models** screens allow you to specify alternative VS Solver libraries. This option is mainly used for custom projects.

13 Checkbox to use alternative VS Solver libraries. Check the box to show a drop-down list 14 and associated fields for specifying vehicle codes and pathnames for the VS Solver libraries 15 (Figure 6).

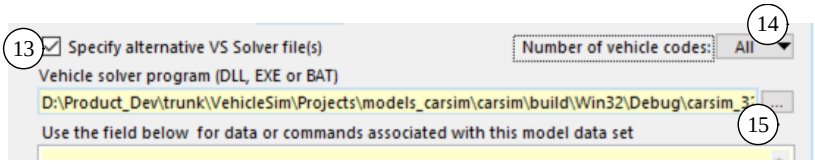


Figure 6. Specify an alternative solver library.

(14) Drop-down list to set the number of vehicle codes in the dataset that are described with a configuration code name and associated pathname for a program file (15). If the solver library is for vehicle models, the first item on the list (**All**) is usually chosen (Figure 6). It is also possible to specify a set of library files paired with codes.

(15) **Vehicle DLL pathname**. When the drop-down control (14) is used to specify **All**, a single field is shown the for the pathname of a VS Solver library DLL or EXE or BAT file.

When the drop-down control (14) is used to specify a number, then pairs of fields are shown, with the first field specifying a model code, and the second specifying the pathname for a VS Solver library. This mode was used for older versions of the software, where multiple solvers were provided that each included a single built-in model. This option is no longer used, and will probably be removed in a future version.

Running Multiple VS Math Models in Parallel

The VS Math Models constructed in CarSim, TruckSim, and BikeSim can run under many external software workspaces, such as Simulink, LabVIEW, and environments that use the Functional Mockup Interface (FMI). When used this way, multiple VS Math Models may be run simultaneously, in parallel. These simulations are set up most easily using the **Parallel VS Math Models** tool, accessed using the **Tool** menu (Figure 7) or the **Libraries** submenu **Batch**.

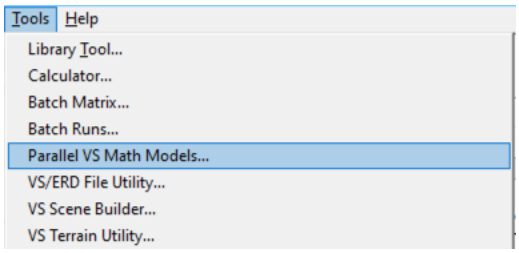


Figure 7. Use the Tools menu to locate the Parallel VS Math Models screen.

Documentation for this tool is available from the Help **Tools** submenu item **Parallel VS Math Models**.

The **Parallel VS Math Models** screen has buttons to access the external workspace (Simulink or an FMI environment) and to refresh the files needed to specify the vehicles and simulation conditions. A checkbox is available on some of the **Models** screens to indicate that multiple VS Math Models will be run (16).

(16) **Run parallel VS Math Models** checkbox. When this box is checked, the **Run Control** dataset(s) that link to this **Models** screen will have the Run button(s) dimmed, to prevent problems of initiating a run without having all of the preparations that are made automatically by the **Parallel VS Math Models** tool.

Models Screens for Simulink and LabVIEW (Windows)

VS models can be extended using Simulink and LabView. The libraries for setting up these extensions are **Models: Simulink** and **Models: LabVIEW for Windows (VI)**, respectively. The screens for these libraries are very similar in operation and function, with the main difference being

that the Simulink screen is set up to work with MDL and SLX project files, while the LabView screen is set up to work with VI project files.

When a VS Solver is combined with Simulink, the vehicle model appears in the Simulink model as an external S-function. When combined with LabView, the vehicle model appears as an external library function.

Running Simulations

Figure 8 shows that when a link is made to either the Simulink or LabVIEW libraries, two buttons appear, along with the link underneath that is used to choose a dataset from the selected library.

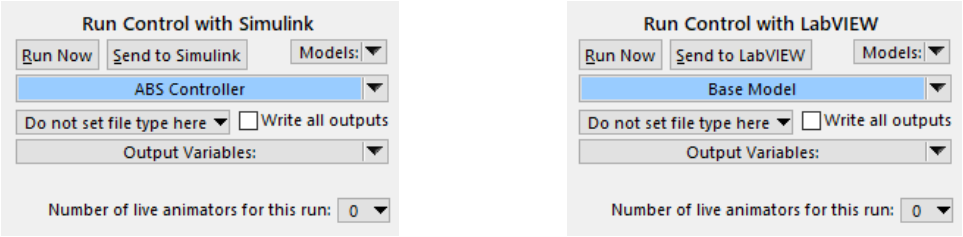


Figure 8. Run Control buttons when linked to a dataset from the Simulink or LabVIEW libraries.

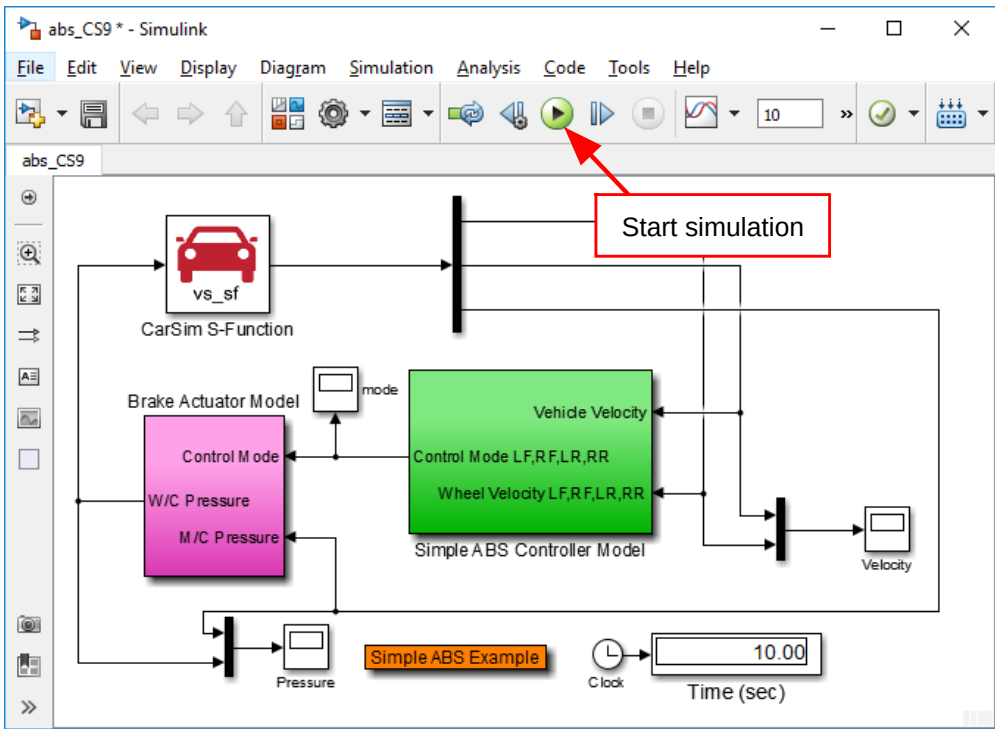


Figure 9. The start button in Simulink.

With either tool, you can initiate a run from the **Run Control** screen using the button **Run Now**. Alternatively, you can control the run from Simulink or LabView. For example, Figure 9 shows the start button in Simulink. When running from Simulink, you should click the button **Send to**

Simulink at least once, to send the necessary data from the VS database to Simulink and the VS Solver. The same is true for LabView, except the button is named **Send to LabVIEW**.

If you are mainly running a VS Browser, then the software uses your active VS database folder (e.g., `CarSim_Data`) as the current working directory and sends commands to Simulink to set the working directory there to be the same. However, if you are mainly running from Simulink, then you can specify an alternative directory and the VS Browser will ensure that the VS Solvers work properly with the Simulink working directory.

LabView for Windows always runs from its own directory. The VS Browser sends commands to LabView using Windows COM, and ensures that the VS Solver works in that environment.

Figure 10 shows the flow of information for the case of Simulink. Use the VS Browser to specify the vehicle model (vehicle properties, road geometry, test conditions, etc.), which is automatically linked to the appropriate VS Solver DLL. A wrapper program (another DLL file) provides the interface between Simulink or LabView and the VS Solver library. During the run, there is close communication between Simulink or LabView and the wrapper DLL, and between the wrapper and the specific VS Solver DLL that is used to construct the VS Math Model for the run. (More detail about the operation is provided in *VS Math Models Reference Manual*.)

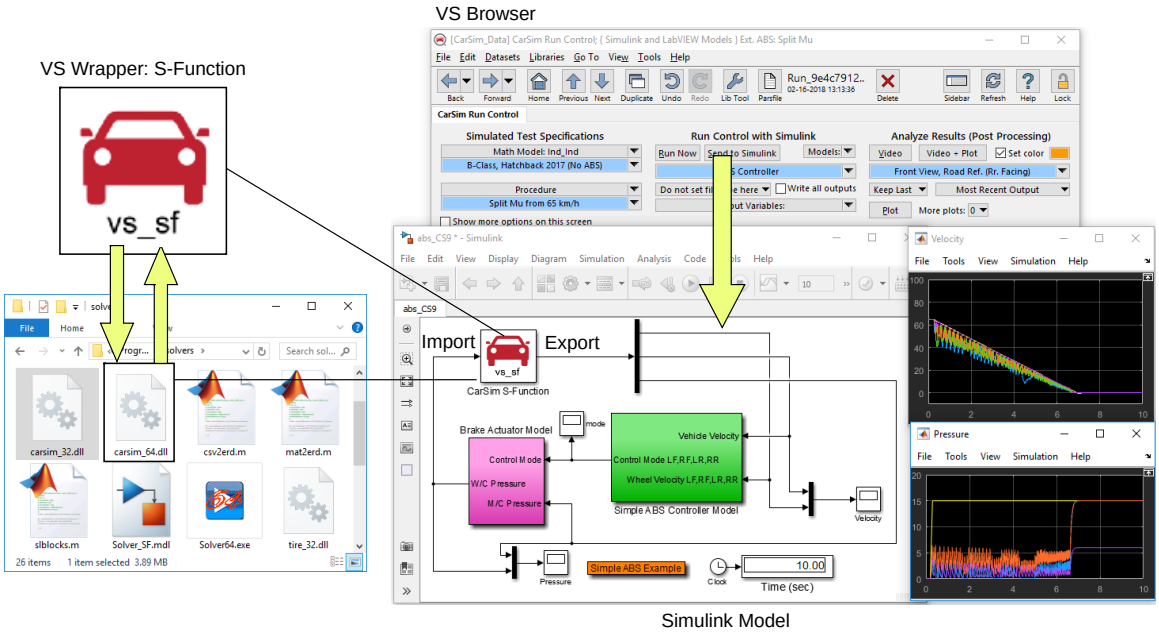


Figure 10. Running a VS Math Model as part of a Simulink model.

Simulink and LabVIEW Models Screen Settings

The descriptions of the common settings for a **Models** screen in the previous sections (see Figure 3 on page 4 and Figure 5 on page 9) were based on the Simulink screen, so only a few notes are provided here.

(11) The external model or program is a Simulink or LabVIEW model file. You will specify the pathname to the MDL or SLX file (Simulink) or VI file (LabView) that will use the VS Solver. The pathname can be typed, pasted in, or obtained using the adjacent browse button.

(12) The Simulink working directory field is visible and active only if the adjacent checkbox is checked (see Figure 5). When the box is not checked, then the working directory is assumed to be the active VS database folder (e.g., `CarSim_Data`) and all pathnames on this screen are shown relative to that.

This field and checkbox do not exist in the LabVIEW screen.

Models: Self-Contained (Windows)

The **Models: Self-Contained Solvers** screen provides a convenient place to integrate with other software to extend models or provide automation on Windows. When a dataset from this library is selected, the **Run Control** screen shows a single button **Run Math Model**. Click on the button to initiate a run that might involve alternative solver libraries or other forms of model extensions. This screen can also be used to associate arbitrary datasets with the VS Solvers, such as a set of VS commands or a numerical integration method.

Nearly all of the discussion in earlier sections about basic and advanced controls applies here (see Figure 3 on page 4 and Figure 5 on page 9). There are just three differences for this screen, which are in the top area (Figure 11).

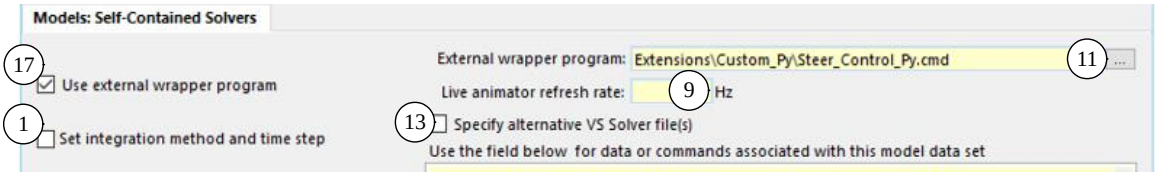


Figure 11. Top of the Models: Self-Contained Solvers screen.

1. There is no working directory.
2. There is no checkbox to **Run parallel VS Math Models**.
2. Instead of specifying a Simulink or LabVIEW model, there is an option to specify a wrapper program (11) that loads the appropriate VS Solver DLL and constructs a VS Math Model using the VS API (see the *VehicleSim API Manual* in the VS SDK for details). This option is enabled by the external wrapper checkbox (17), which only exists on this screen.

VS Math Models are normally run from the VS Browser. However, in support of automation and custom environments, each product includes simple command-line programs called VS Solver Wrappers that can be used to load the VS Solver libraries and run VS Math Models.

The products include both 32- and 64-bit solver libraries, e.g., `carsim_32.dll` and `carsim_64.dll` for CarSim. A separate wrapper is needed for each: `VS_SolverWrapper_CLI_32.exe` and `VS_SolverWrapper_CLI_64.exe`. Both are located in the `Programs` folder for the product, and are documentated in the Tech Memo *VS Solver Wrapper*.

There may be occasions to a VS Solver Wrapper, such as to run the 64-bit version of the VS Solver library (the Browser is a 32-bit program that can only load 32-bit DLL libraries directly. In these cases, the wrapper can be specified on this screen (11) (Figure 12). When a VS Solver Wrapper is

specified, and additional checkbox is shown (18), which only exists on this screen. Normally, a terminal window appears when the simulation is running, and closes when the run completes.

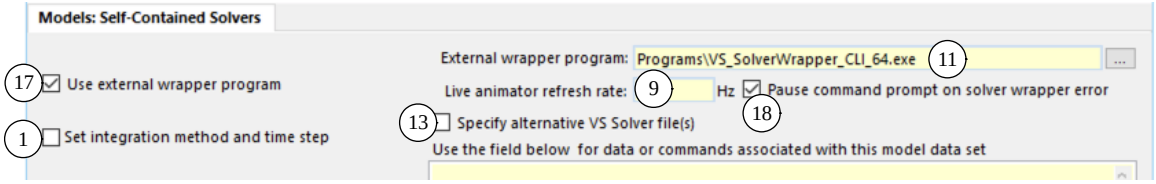


Figure 12. Extra checkbox when wrapper is VS_SolverWrapper_CLI.

If the box (18) is checked and an error occurs making a simulation, the terminal window remains visible with error information. If there is no error, the window closes when the run completes, even if the box is checked.

Models: Transfer to Local Windows Directory

The **Models: Transfer to Local Windows Directory** screen is used to control transfer of files between a VS database and other software running in a directory accessible through Windows. It is mainly used for advanced projects in which a VS product is used with external software that requires control of the Windows working directory. For example, VS Math Models can be extended using external software such as Simulink or automated using software such as MATLAB or Visual Basic.

Figure 13 shows that when a dataset from this library is selected on the **Run Control** screen, a single button is shown: **Generate Files for this Run**. Click this button to generate the files needed run the simulation. (For details about the input and output files, see *VS Math Models Reference Manual*.) The dataset is set up so that output files are placed in the VS database for convenient post-processing plotting and animation.

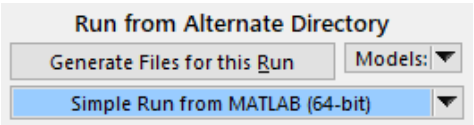


Figure 13. Run control button when linked to a dataset from the Models: Transfer library.

This screen is nearly the same as the **Models: Self-Contained Solvers**, except that the main purpose of the dataset is to specify an external directory rather than a wrapper program.

Nearly all of the discussion in earlier sections about basic and advanced controls applies here (see Figure 3 on page 4 and Figure 5 on page 9). There are just three differences for this screen, which are in the top area (Figure 14).

1. There is no external program or project specified to construct and run the VS Math Model. This work is done remotely, using only the data files that were transferred from the database.
2. There is no checkbox to **Run parallel VS Math Models**.
3. A drop-down control (17) is used to specify whether the external program is a 32-bit application or a 64-bit application. This control only exists on this screen.

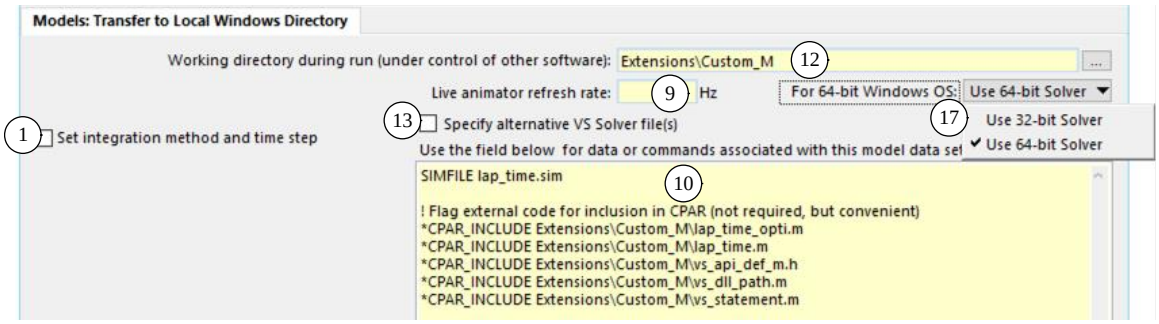


Figure 14. Top of the Models: Transfer to Local Windows Directory screen.

The simfile that is written to the working directory includes a pathname for the VS Solver library. Given that the Windows versions of VehicleSim products include both 32-bit and 64-bit solver libraries, this screen includes a drop-down control (17) to specify which one should be identified in the Simfile. However, if the pathname for the VS Solver DLL is set explicitly (when the box (13) is checked), then the specified DLL is used and the 32/64 drop-down control is hidden.

Models: Export FMU/FMI

This screen is used to export a VS Solver library to a Functional Mockup Unit (FMU) that connects as a "slave" (client) of a co-simulation for a Functional Mockup Interface (FMI) "master" (host) to run on the Windows or Linux environments. You can integrate this FMU into other simulation environments, for example MATLAB/Simulink with PSP, dSPACE VEOS and so on.

Run Control

Figure 15 shows that when a dataset from this library is selected, the default button is replaced with a single button to generate the FMU.

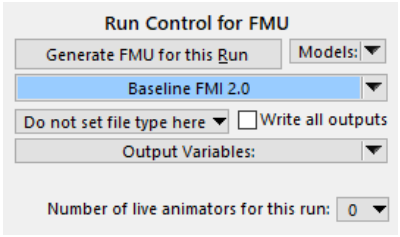


Figure 15. Run control buttons when linked to a dataset from the FMU/FMI library.

When you click the **Generate FMU for this Run** button, the VS Browser generates an FMU for this model and the supporting datasets. You can then switch to the external environment to manage the simulation using the new FMU.

Models Screen Settings

Nearly all of the discussion in earlier sections about basic and advanced controls applies here (see Figure 3 on page 4 and Figure 5 on page 9, which document the controls numbered (1) - (16)).

This screen adds controls (18) - (25) that support the creation of FMUs (Figure 16); they are described below.

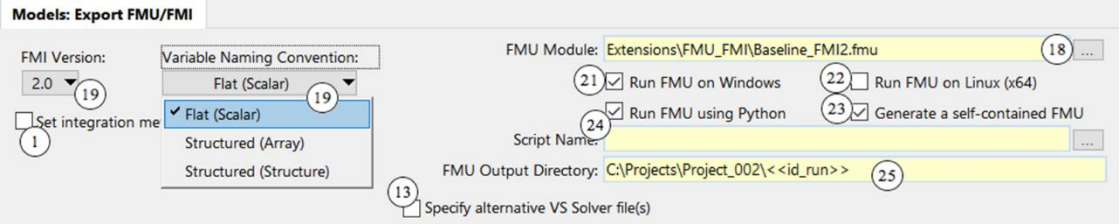


Figure 16. Top of the Models: Export FMU/FMI library screen.

18 Pathname for the FMU file that will be created.

19 Drop-down control to specify the FMI version. FMI versions 1.0 and 2.0 are supported. The version should match the requirements of the FMI master software tool.

20 Drop-down control to specify a naming convention: "Flat," "Structured (Array)," or "Structured (Structure)." The differences between the three depend on the FMU master software. In a Simulink interface, the three options relate the import/output "signals" to "ports" as follows:

- Flat: each import/export variable is represented with a separate port with one signal (see the "Flat "example in Figure 17).

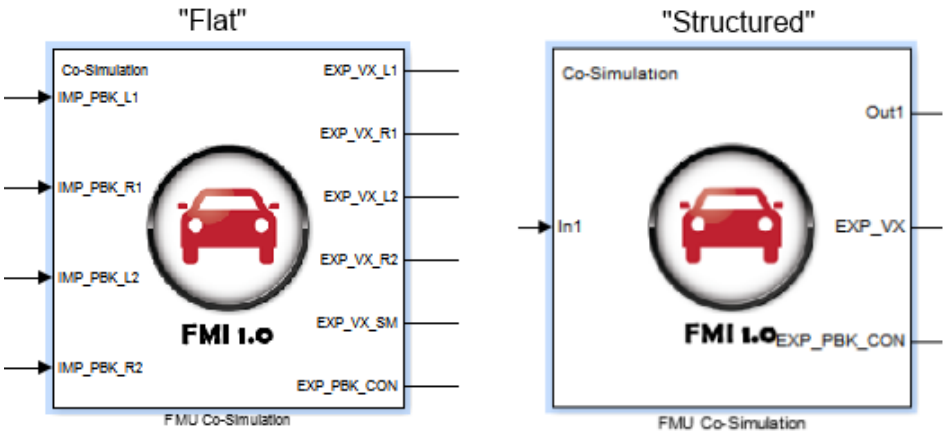


Figure 17. Variable Name Conventions "Flat" and "Structured (Structure)"

- Structured (Array): all specified import variables are handled with a single port with all imports included as signals, and all export variables are also handled with a single port with all exports included as signals.
- Structured (Structure): the import and export variables are organized into multiple ports, where each port can handle one or more signals (see the Structured" example in Figure 17).

21 Checkbox to create the FMU to run on the Windows system environment.

22 Checkbox to create the FMU to run on the Linux 64-bit system environment, not Real-Time Linux. The Linux FMU supports FMI 2.0 only.

You must select **Run FMU on Windows**, **Run FMU on Linux (x64)**, or both.

(23) (25) Checkbox to generate a self-contained FMU. If this checkbox is unchecked, the FMU uses the VS Browser database, and simulation results will be written in the database `Results` folder in a similar fashion to simulations made with built-in models. In this case, you can use VS Visualizer to view the results.

If this checkbox is checked, the FMU contains everything needed to run the simulation: a Simfile, a Parsfile, External tire model/data/road files, and the VS Solver.

Note If you select "Run FMU on Linux (x64)," checkbox (23) is automatically checked and FMI 2.0 (19) is selected.

Note Starting with version 2021.1, you can include an external tire model in an FMU. The FMU must be generated with FMI 2.0 and will only run on a Windows system. If the license of the external tire model is required, please contact Technical Support for help.

For FMI 1.0, the FMU master program extracts the FMU to a folder that contains a subfolder named "sources" (FMI 1.0). The results are saved in the "sources" folders.

For FMI 2.0, the path field "FMU Output Directory" will be available for a self-contained FMU, and you can specify the output location, for example:

On Windows `C:\My_FMU_Output\`
On Linux `~/My_FMU_Output/`

The symbol "~" represents user's "Home" folder. On Linux it is `/home/{user}/`, and on Windows it is `%USERPROFILE%\Documents\`. If this field is blank, by default the FMU will write output to:

`%USERPROFILE%\Documents\{Product name}_results\{version}\<<id_run>>` on Windows.

`~/{Product name}_results/{version}/<<id_run>>` on Linux.

You can use the "`<<id_run>>`" macro to insert the Run Control's UUID. You can use back slash "\" or forward slash "/" as the folder delimiter on Windows and Linux platform. The FMU will convert it automatically.

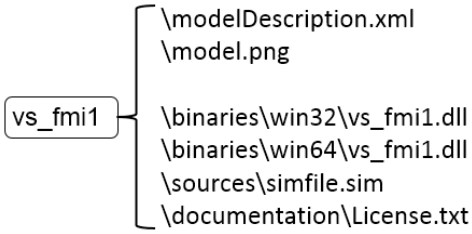
On Windows platform, if you do not specify the output directory and if access to the default location, `%USERPROFILE%\Documents\`, is denied, FMU will write the results in the resources folder of unpacked FMU. The unpacked FMU folder is created by FMU master software.

(24) Checkbox to launch a Python script as FMU master and path name for the Python script file. If you have Python 2.7 installed, this checkbox will be enabled. You must have PyFMI installed to run a Python master script.

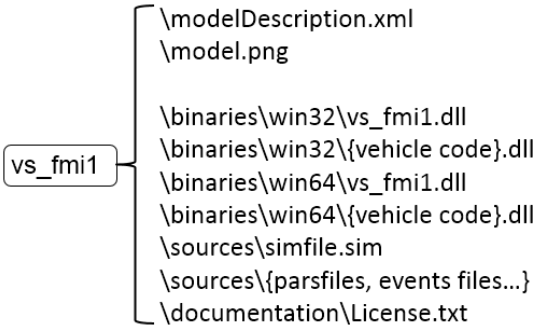
FMU Content

The FMU generated by the VS Browser contains the following structures:

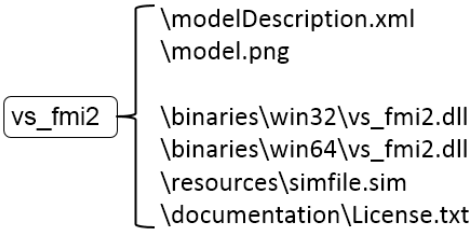
(1) FMI 1.0 Shared data:



(2) FMI 1.0 Self-contained:



(3) FMI 2.0 Shared data:



(4) FMI 2.0 Self-contained:

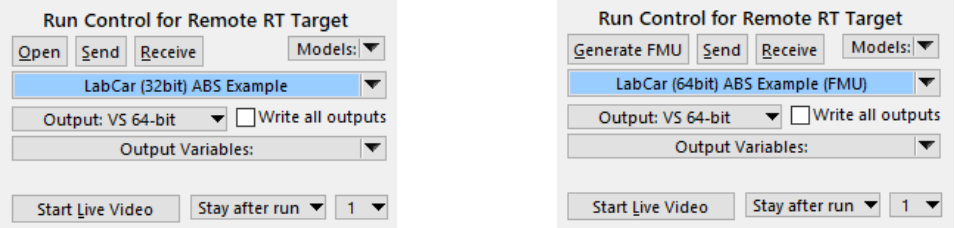
Models: Transfer to Remote RT Target

This library is used to store information needed for real-time HIL versions of VS products where the software is distributed over two or more computers and the VS Math Model is run with an external simulation tool. Some external tools supported by this library are ETAS LabCar, Concurrent SimWB, and A&D. When a link is made to a dataset from this library, the VS Browser manages the transfer of files from with Windows host to and from another target machine.

The datasets support two methods for connecting the VS Solver to the external simulation workspace. One is the Simulink S-Function compiled with Simulink Coder (formerly known as Real-Time Workshop) for the target OS. The other is a Functional Mockup Unit (FMU) that supports the Function Mockup Interface (FMI), as described in the previous section.

Run Control

When this library is selected, the default Run button on the **Run Control** screen is replaced with three buttons and controls for live video are shown (Figure 18).



a. Link to a dataset with a Simulink model. b. Link to a dataset with an FMU definition.

Figure 18. Run control buttons when linked to a dataset from the Transfer by FTP library.

As shown in the figure, the buttons shown depend on whether the linked dataset identifies a Simulink model (Figure 19a) or specifications for an FMU (Figure 19b).

If the linked dataset identifies a Simulink model, there is a button labeled **Open**. When you click it, you are transferred to Simulink, to view and edit the specified Simulink model.

If the linked dataset specifies an FMU, there is a button labeled **Generate FMU**. When you click this button, the VS Browser creates the FMU using the current simulation data.

When you click the **Send** button, a set of Parsfiles with all data needed by the VS Solver for the run is created and sent to a target location specified in the linked dataset. This also initializes live video if it has been enabled.

To run the simulation, you need to switch to the specified tool (LabCar, SimWB, etc.) and use the external tool to manage the building of the model and running it on the target machine.

To work with real-time systems using Simulink, the VS Solver is provided as a library file prepared for the specific target system (e.g., LinuxRT). The VS Solver is already compiled to provide the library file, and is never recompiled for use in a new run. However, access to the VS library comes through your Simulink model, which must be recompiled if any changes are made to the Simulink model extensions, such as the selection of Import or output variables, time step, and other simulation properties.

When you click the **Receive** button, all output files generated in the simulation run are transferred from the target to the host for post-processing animation and plotting. This action is usually not needed; the transfer is done automatically by the VS Browser if you run only one VS Solver on the target machine. (The VS Solver signals to the VS Browser when a simulation run has finished.) Depending on the status of a checkbox (22) (Figure 19), the files on the target machine might be deleted automatically after the transfer.

Models Screen Settings

Nearly all of the discussion in earlier sections about basic and advanced controls applies here (see Figure 3 on page 4 and Figure 5 on page 9, which document the controls numbered (1) - (16). The pathname for a generated FMU file (18) was introduced for the FMU library screen.

This screen adds controls (22) - (27) that specify the protocol and settings for transferring data between the Windows host machine and the RT target machine (Figure 19). It also adds a drop-down control (28) for specifying the Simulink model or FMU type.

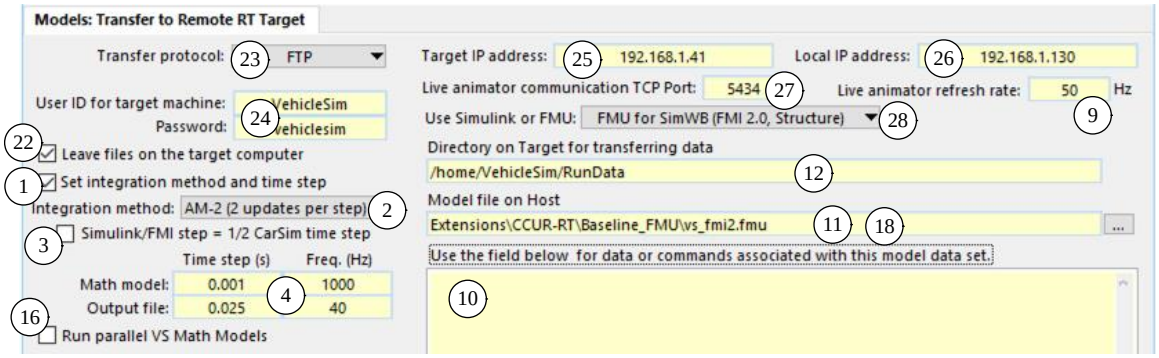


Figure 19. Top of the Models: Transfer to Target by FTP library screen.

The target directory (12) is used when you click the **Send** and **Receive** buttons from the **Run Control** screen. The pathname must be valid for the target machine, and is typically UNIX-style (with '/' delimiters and no spaces).

Note that a single field is used either for specifying a Simulink model (11) or pathname for a FMU file that will be created (18), with the interpretation being based on the setting of the drop-down control (28).

Data transfer controls ((22) and higher) are described below.

(22) Checkbox for the option to leave files on the target computer. When a run is made on the target computer, output files might be generated that can later be transferred to the host (the host is the Windows machine running the VS Browser) for animation, plotting, and other analyses.

When this box is checked, the only action taken when you click the **Receive** button on the **Run** screen is that the files are copied. When the box is not checked, the files are deleted from the target after they are transferred.

The delete/leave alone behavior also applies when files are moved automatically by the VS Browser when the VS Solver signals that the simulation has finished.

(23) Drop-down control with options for the transfer protocol used to move files to and from the target machine (Figure 20).

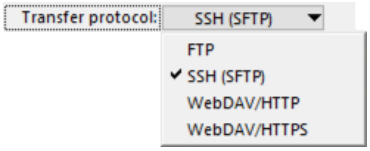


Figure 20. Transport protocols available for use with RT targets.

(24) User ID and password for target machine.

(25) IP address of target machine where the VS Math Model runs. This is needed for real-time animation, to allow one or more live animators to seek connections with a running VS Math Model.

(26) Local IP address (host machine) that connects to the real-time target. This is needed for real-time animation, to allow the VS Math Model running on a target machine to communicate with the animator, running on the host machine.

(27) Live animator communication TCP port.

(28) Drop-down control to select the external modeling tool. Use this to choose among four supported tools (Figure 21). If the Simulink option is selected, then the project pathname (11) should be a Simulink model file. If an FMU option is selected, then the project pathname should be the name of the FMU file that will be generated (18).

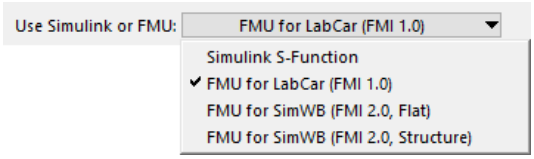


Figure 21. Drop-down control to select external modeling tool.

Models: Transfer to NI-RT Target

Datasets from the **Transfer to NI-RT Target** library support two real time software environments from NI: LabVIEW-RT and VeriStand. In these systems, the software is distributed over two or more computers and the VS Math Model is run as a LabVIEW/VeriStand library function.

Multiple VS Math Models can run at the same time, if the LabVIEW model is built with more than one reference to the VS Solver library function. However, this does not work in VeriStand.

Run Control

Figure 22 shows that when this library is selected, the default button is replaced with four buttons. Controls also appear for controlling live video.

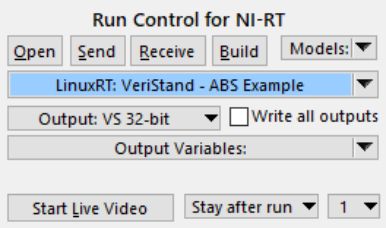


Figure 22. Run control buttons when linked to a dataset from the LabVIEW RT library.

When you click the **Open** button, you are transferred to LabVIEW/VeriStand to view and edit the specified LabVIEW/VeriStand model, and run the model on NI Real-Time target.

When you click the **Send** button, a set of Parsfiles with all data needed by the VS Math Model constructed for the run is created and sent to the target location. If live animation is specified and the animator is not already running, then it is launched at this time.

When you click the **Receive** button, all output files generated in the simulation run are transferred from the target to the host for post-processing animation and plotting. This action is usually not needed; the transfer is done automatically if you run only one VS Solver on the target machine.

Models Screen Settings

Nearly all of the discussion in earlier sections about basic and advanced controls applies here (see Figure 3 on page 4 and Figure 5 on page 9, which document the controls numbered (1) - (16). The pathname for a generated FMU file (18) was introduced for the FMU library screen. Controls (22) - (27) that specify the protocol and settings for transferring data between the Windows host machine and the RT target machine were described in the preceding section (see Figure 19 on page 20).

Figure 23 shows the top of the **Transfer to NI-RT Target** with the controls numbered for compatibility with earlier sections. New controls are described below.

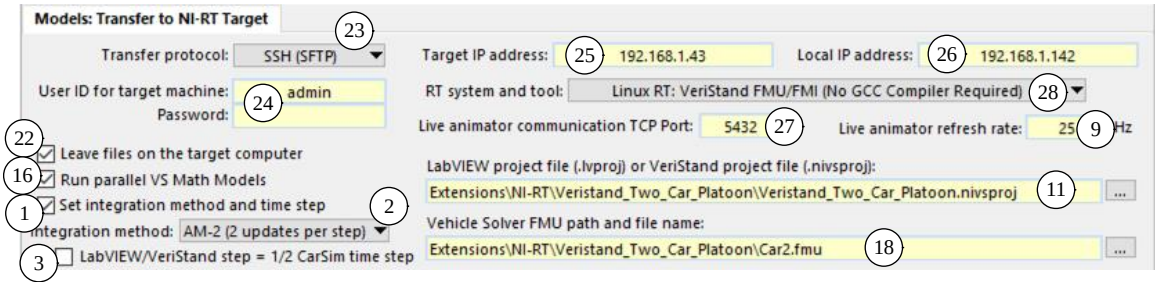


Figure 23. Top of the Models: Transfer to NI RT Target library screen.

(28) Drop-down control to choose among four supported NI target types and tools (Figure 24). This selection determines the type of NI project file that is specified (11).

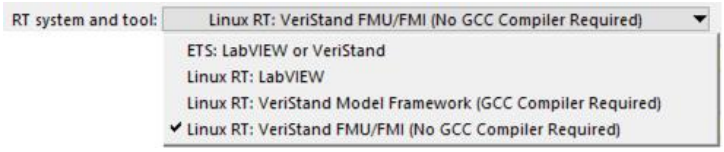


Figure 24. Choices for NI system and tool.

Depending on the selection, another field might also be shown:

1. **ESL LabVIEW or VeriStand**: no field is shown, i.e., (18) is hidden.
2. **Linux RT: LabVIEW**: no field is shown, i.e., (18) is hidden.
3. **Linux RT: VeriStand Model Framework**: a field is shown for the pathname of the CGG compiler (29) (Figure 25).
4. **Linux RT: FMU/FMI**: a field is shown for the pathname of the FMU file that will be created (18) (Figure 23).

(29) Pathname for the GNU C/C++ compiler. This field is shown only for the third option of the drop-down control (28). This control only exists on this screen.

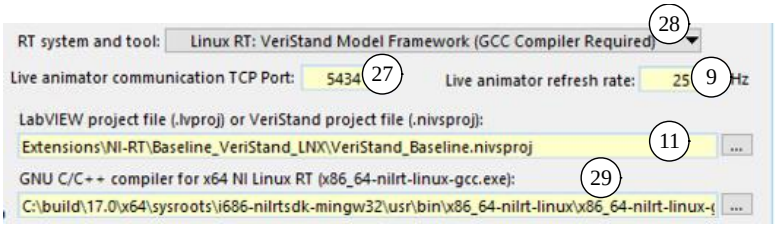


Figure 25. Appearance of the NI RT Target screen when the GCC compiler is needed.

Models: Transfer to RT-Lab version 10+ Target

This section describes the screen used to set up runs using one or more target machines running with RT-Lab software (versions 10 and higher) from Opal-RT.

Run Control

Two basic actions are needed to make a run on the RT-Lab v10+ target machine:

1. Build the solver by compiling the Simulink model and linking to the VS Solver library file. This step is needed only if changes were made to the Simulink model.
2. Download the compiled Simulink model and a set of Parsfiles to the target machine.

If live animation is enabled, a third action is added:

3. Launch the animator on the Windows host. The animator will wait until the VS Solver establishes a connection.

Figure 26 shows that when a link is made to a dataset from this library, the default button is replaced with three buttons and controls appear for controlling live video.

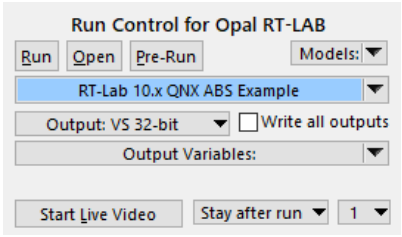


Figure 26. Run control buttons when linked to a dataset from the RT-Lab v10+ library.

When you click the **Run** button, the VS Browser sends files from the Windows host to the RT-Lab target, makes the run, and then copies output files back to the host for post-processing animation and plotting.

When you click the **Open** button, you are transferred to Simulink to view and edit the specified Simulink model.

When you click the **Pre-Run** button, the VS Browser refreshes the Parsfiles used for the next run.

Any start time setting on the **Procedure** or **Run Control** screen will be ignored, because the RT-Lab software always sets the start time to zero.

Models Screen Settings

Nearly all of the discussion in earlier sections about basic and advanced controls applies here (see Figure 3 on page 4 and Figure 5 on page 9, which document the controls numbered (1) - (15). Controls (24) - (26) that specify the protocol and settings for transferring data between the Windows host machine and the RT target machine were described earlier (see Figure 19 on page 20).

Figure 27 shows the top part of the **Models: Transfer to RT-Lab version 10+ Target** screen with the controls numbered for compatibility with earlier sections. A few new controls are described below.

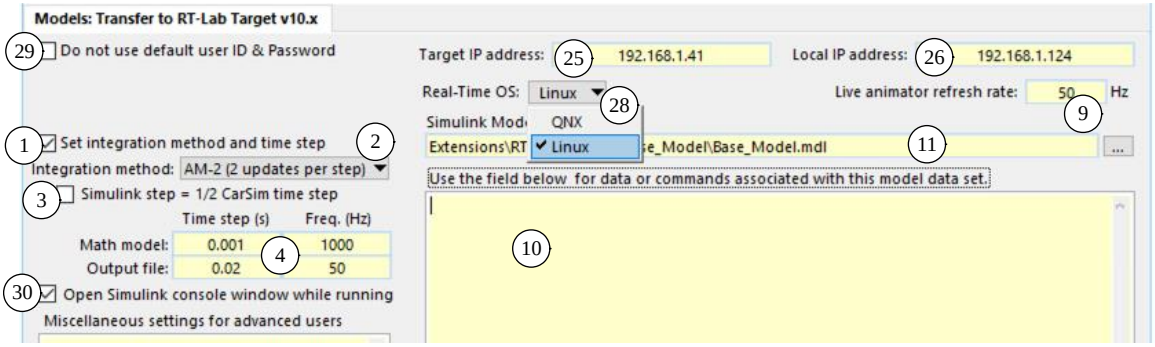


Figure 27. Top of the Models: Transfer to RT-Lab v10+ Target library screen.

(28) On this screen, the options specify the RT operating system. There are two options: QNX and Linux (32-bit).

(29) Checkbox to disable defaults and specify a custom ID and password for the target machine. If checked, fields are shown for user ID and password. This control only exists on this screen.

(30) Checkbox to enable display the Simulink host subsystem during real time running. This control only exists on this screen.

This screen does not have the checkbox for running parallel VS Math Models nor the option to leave files on the target computer.

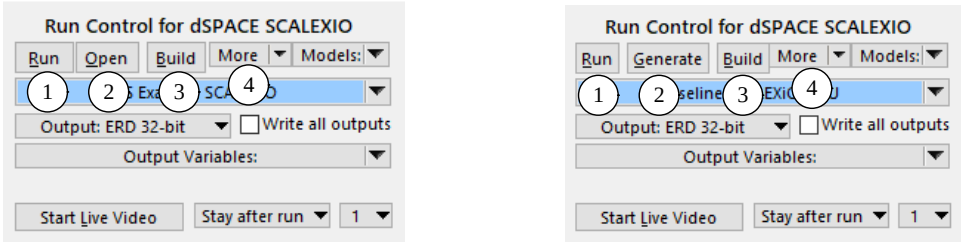
Models: Transfer to dSPACE SCALEXIO Target

This section describes the screen used to set up runs for a dSPACE SCALEXIO real-time target. The datasets support two methods for connecting the VS Solver to the external simulation workspace. One is the Simulink S-Function compiled with Simulink Coder (formerly known as Real-Time Workshop). The other is a Functional Mockup Unit (FMI) that supports the Function Mockup Interface (FMI).

Alert If the pathname of the location of your programs folder contains blank space, for example `C:\Program Files\CarSim_Prog`, Simulink Coder will not create target execution code.

Run Control

When you are linked to a dataset in the **Models: Transfer to dSPACE SCALEXIO Target** library, four controls are shown for communicating with the dSPACE system (Figure 28).



a. Link to a dataset with a Simulink model. b. Link to a dataset with an FMU definition.

Figure 28. CarSim Run Control buttons for dSPACE SCALEXIO.

As shown in the figure, the buttons shown depend on whether the linked dataset identifies a Simulink model (Figure 28a) or specifications for an FMU (Figure 28b).

(1) **Run** button. When you click this, the VS Browser will send the Parsfiles to the target, start the animator if it is enabled, and start `RunScalexio.exe`, which will run a python script to start ControlDesk NG, open the project associated with this dataset, reload variable configurations, and then start online calibration and measurement as the application runs on the target.

(2) If the linked dataset identifies a Simulink model, the second button is labeled **Open**. When you click it, the VS Browser will open MATLAB and the Simulink model specified in the dataset and set the path in MATLAB.

On the other hand, if the linked dataset has specifications for an FMU, the button is named **Generate**. When you click it, the VS Browser will generate the FMU.

(3) **Build** button When you click this, the VS Browser will start up ConfigurationDesk, open the ConfigurationDesk project associated with the dataset, and build the model using ConfigurationDesk with the Simulink model or FMU.

(4) Drop-down control with more options ((5) - (8), Figure 29).

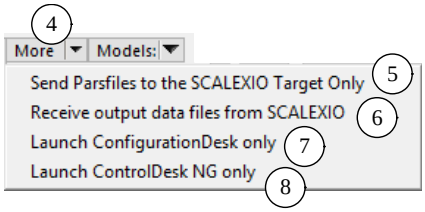


Figure 29. More options for a SCALEXIO connection.

(5) **Send Parsfiles to the SCALEXIO Target Only**: send the Parsfiles and all other required files to the target machine.

(6) **Receive output data files from SCALEXIO**: retrieve the relevant files from the target as of the latest run. This is automatically done at the end of each run on the target, but the files will remain on the target and can be manually retrieved again later using this option

(7) **Launch ConfigurationDesk only**: start up ConfigurationDesk and open the project associated with this dataset.

(8) **Launch ControlDesk NG only**: open ControlDesk NG and the project associated with this dataset.

Models Screen Settings

Nearly all of the discussion in earlier sections about basic and advanced controls applies here (see Figure 3 on page 4 and Figure 5 on page 9, which document the controls numbered (1) - (16). Controls (24) - (26) that specify the protocol and settings for transferring data between the Windows host machine and the RT target machine were described earlier (see Figure 19 on page 20).

Options to specify alternative solver libraries do not exist on this screen, nor is there an option to specify an alternate working directory.

Figure 30 shows the **Models: Transfer to dSPACE SCALEXIO Target** screen with the controls numbered for compatibility with the discussion in previous sections. In this example, a Simulink model is used to connect with the VS Math Model (11), based on the status of the checkbox (28).

Figure 31 shows that when the checkbox (28) is checked to specify that an FMU will be generated, new controls are shown ((34) and (35)), and the Simulink field is replaced with a field is used to specify a pathname for the FMU (18).

The new controls that exist only for dSPACE are described below.

(31) Drop-down control to specify whether the target machine has QNX or Linux RT operating system.

(32) ConfigurationDesk project folder.

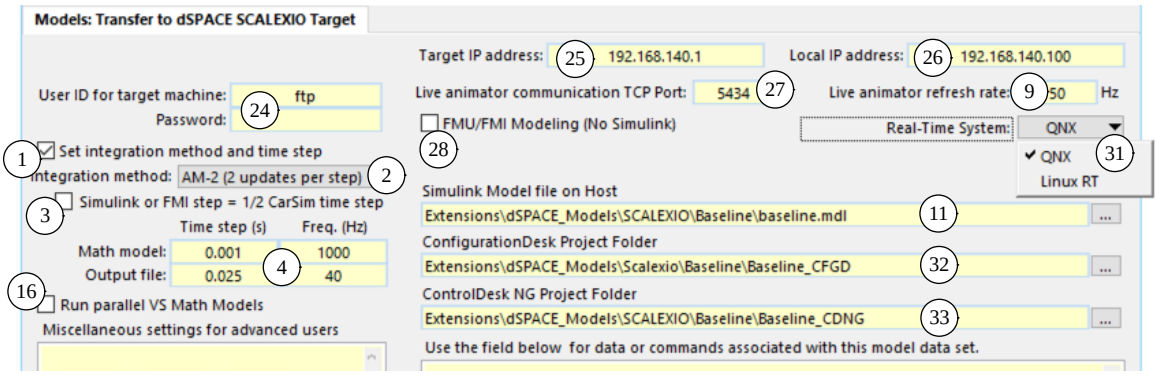


Figure 30. Top of the SCALEXIO screen for an example dataset using a Simulink model.

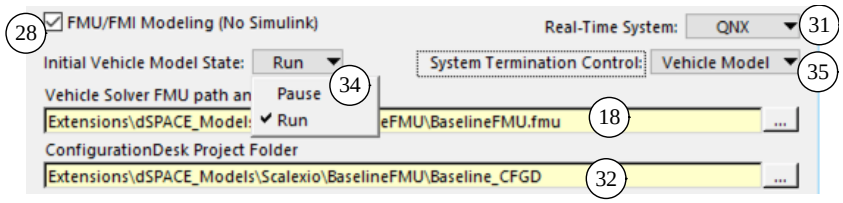


Figure 31. Part of the SCALEXIO screen for an example dataset that makes an FMU file.

33 ControlDesk NG project folder.

> **Note** If you are creating a new dataset rather than using a pre-existing example, you need to first create the ConfigurationDesk and ControlDesk NG projects before setting their locations.

34 Drop-down control to specify the initial state of the VS Math Model. If a lengthy initialization is required, the **Pause** option will prevent the vehicle math model from getting ahead of the rest of the simulation. Otherwise, the **Run** option will start immediately.

35 Drop-down control to specify how the simulation will be terminated (Figure 32).

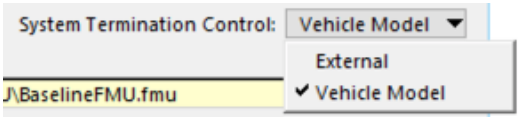


Figure 32. Options for terminating a simulation.

Models: Transfer to dSPACE Target

This section describes the screen used to set up runs for a dSPACE DS1005, DS1006, or DS1103 real-time target using Simulink Coder (formerly Real-Time Workshop).

Run Control

Overall, three actions are needed to make a run on the dSPACE target:

1. Compile the Simulink model and link to the VS Solver library file. This step is needed only if changes were made to the Simulink model. This can take anywhere from 20 seconds to several minutes, depending on the complexity of the Simulink model.
2. Download the compiled Simulink model to the dSPACE board. This step is needed for the first run to be made with a given Simulink model. It typically takes about half a minute.
3. Download a set of Parsfiles with all data needed by the VS Solver for the run. This step is needed every time a setup run is changed, or when any setting from the VS GUI is changed. It typically takes a few milliseconds and is hard to notice.

If live animation is enabled, a fourth action is added:

4. Launch VS Visualizer on the Windows host. VS Visualizer will wait until the VS Solver establishes a connection.

> **Note** If the number of live animations is not zero, and if Live Animation is not already running, then the VS Browser will launch Live Animation. Otherwise, the VS Browser will skip this step.

When you are linked to a dataset in the **Models: Transfer to dSPACE Target** library, several controls are shown for communicating with the dSPACE target (Figure 33).

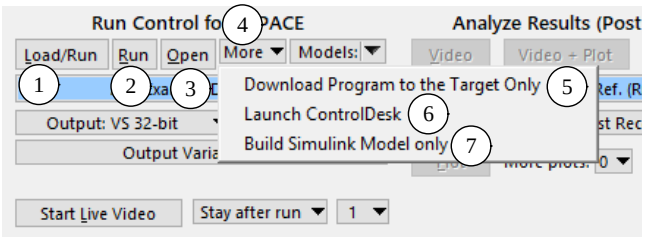


Figure 33. Run control buttons when linked to a dataset from the dSPACE library.

(1) When you click the **Load/Run** button, the VS Browser will:

1. launch Live Animation if needed for live animation;
2. download the compiled Simulink model to the dSPACE target board;
3. load dSPACE ControlDesk software with the project identified on the **Models: Transfer to dSPACE Target** screen, if this is enabled;
4. generate one or more Parsfiles with all information needed by the VS Solver, and download the Parsfile(s) to the target board; and
5. start the simulation.

(2) When you click the **Run** button, the VS Browser will:

1. launch Live Animation if needed for live animation;
2. generate one or more Parsfiles with all information needed by the VS Solver, and download the Parsfile(s) to the target board; and

3. start the simulation.

(3) When you click the **Open** button, the VS Browser will:

1. open the Simulink model identified on the **Models: Transfer to dSPACE Target** screen, and
2. set the MATLAB path and environment.

(4) The **More** drop-down control provides three specialized options:

(5) **Download Program to the Target Only.** Download the compiled Simulink model to the dSPACE target board.

(6) **Launch ControlDesk.** Load dSPACE ControlDesk software with the project identified on the **Models: Transfer to dSPACE Target** screen.

(7) **Build Simulink Model Only.** Compile the Simulink model and link to the VS Solver library file.

Any start time setting on the **Procedure** or **Run Control** screen will be ignored, because the dSPACE software always sets the start time to zero.

The RundSPACE Utility Program

A Windows utility program named **RundSPACE** controls the simulation process. This utility is shown in Figure 34 and located in `Programs\RundSPACE.exe`.

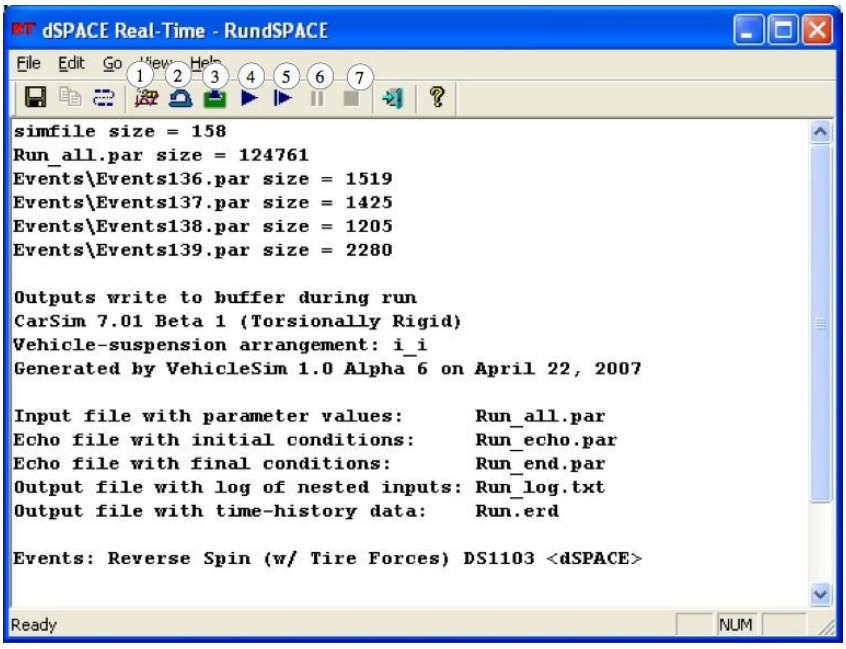


Figure 34. Utility program `RundSPACE.exe`

This program handles the download of the Parsfile(s), controls the program running on the dSPACE board, uploads results back to the host, and communicates with the live animator. This program must remain active for the entire duration of the simulation process.

The program has a row of buttons, shown in Figure 34 and described below.

(1) Launch dSPACE ControlDesk.

(2) Launch Live Animation (depending on settings), download the compiled Simulink model, download the existing expanded Parsfile, and start the simulation. (This is almost the same as the **Load/Run** button (see Figure 33) except it does not recreate the Parsfile.)

(3) Download the current Parsfile to the dSPACE target board.

(4) Run the simulation as currently set up on the dSPACE target board.

(5) Download the current Parsfile to the dSPACE target board and start the run. (This is the same as clicking the buttons (3) and (4) in sequence.)

(6) Pause the simulation.

(7) Stop the simulation.

Writing Outputs for Post-Processing

Outputs from simulation runs made with a dSPACE target machine can be handled two ways for post-processing animation, plotting, and other analysis.

1. The outputs can be placed in a buffer in memory during the run. When the run ends, the buffer with all output variables is sent to the Windows host, where it is written to file.
2. The outputs can be streamed from the dSPACE target to the Windows host continuously as the run proceeds, where the host copies them to file.

The choice is selected with a checkbox on the **Models: Transfer to dSPACE Target** screen (17) (Figure 35, page 31).

When possible, the first method is preferred (with the box (17) checked). A problem with the second method (box unchecked) is that there is limited bandwidth between the host and target, and it is possible to lose data when attempting to transfer too much.

A potential problem with the memory buffer (box checked) is that the amount of data collected during a run might exceed the available memory. The memory required in bytes is: (number of written channels) × (run time in seconds) × (output frequency) × (bytes per output variable). The memory is equal to the size of the BIN file that will be generated after the run is complete. You can look through the `Runs` folder in your database (e.g., `CarSim_Data`) to see the sizes of existing BIN files (most are less than 5 MB).

The amount of available memory depends on the dSPACE hardware board: DS1006 and DS1005 have 128MB and DS1103 has 96MB.

When the stop condition is set to “run forever” or you would like to run for a very long time, the memory might be an issue. The simplest option is to set a specific size for the buffer using the keyword `NSAMP_BUFFER`. For example, the line

```
NSAMP_BUFFER 1000
```

sets the buffer to hold 1000 samples. The number of saved values is `NSAMP_BUFFER` × `NOUT_WRITE`, where `NOUT_WRITE` is the number of output variables. If the number of samples

in the run exceeds NSAMP_BUFFER, then only the most recent NSAMP_BUFFER samples are saved.

Another option to control the amount of required memory is to actively enable and disable the saving of output data during the run using VS Events or equations added with VS Commands. Use the keyword OPT_WRITE to enable writing (OPT_WRITE = 1) or disable it (OPT_WRITE = 0). When disabled, no memory is consumed.

Vehicle Configurations and VS Solver Libraries

When running on a real-time dSPACE system, the VS Solver is provided as a compiled library file prepared for the specific dSPACE hardware (DS1006, DS1005, etc.). The library file is accessed through a Simulink model, which is in turn compiled using a dSPACE makefile (dsmake). Although the VS Solver is already compiled, your Simulink model must be recompiled if any changes are made to the Simulink model extensions, such as the selection of Import or output variables, time step, or other simulation properties.

Models Screen Settings

Nearly all of the discussion in earlier sections about basic and advanced controls applies here (see Figure 3 on page 4 and Figure 5 on page 9, which document the controls numbered (1) - (15).

Figure 35 shows the top part of the **Models: Transfer to dSPACE Target** screen with the controls numbered for compatibility with earlier sections. Controls for specifying the numerical integration options, the datasets for Import and Export variables, and VS commands are the same as with other screens.

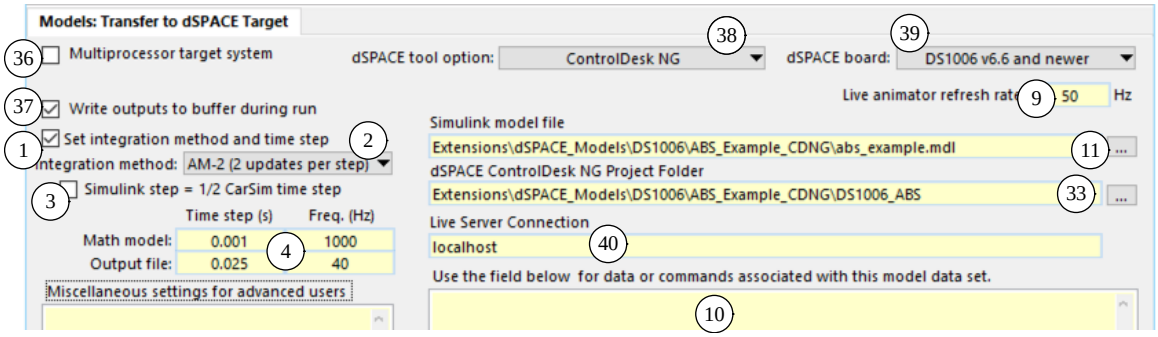


Figure 35. The top part of the Models: Transfer to dSPACE Target screen.

Options to specify alternative solver libraries do not exist on this screen, nor is there an option to specify an alternate working directory, nor an option to run parallel models.

As with the Simulink screen, there is a field for specifying a Simulink model (11), which is compiled for the target machine under the control of dSPACE.

Alert	If the pathname of the location of your programs folder contains blank space, for example C:\Program Files\CarSim_Prog, and your dSPACE product release is earlier than version 6.0, the dSPACE builder will generate an error when you link to the VS Solvers library. To avoid

the problem, you can type `*USE_SHORT_PATHNAME 1` into the miscellaneous field (10).

Additional controls are described below.

(33) ControlDesk experiment project CDX file or folder. This establishes the connection to the Simulink model running on the target machine, based on the selected tool option (38). If the dSPACE ControlDesk NG took is specified, this is the project folder. Otherwise, it is a pathname.

(36) Checkbox and field for specifying the board to use in a multiprocessor target system. When checked, you must specify the dSPACE board name. (Obtain this name with the dSPACE tool ControlDesk). Do not check if there is only one dSPACE processor board available.

(37) Checkbox for option to write outputs to a buffer in memory during the run (keyword = `OPT_BUFFER_WRITE`), as described in the previous subsection.

When checked, output variables are transferred back to the host after each run terminates. This requires memory to save the output values during the run. It does not affect real-time animation.

When unchecked, outputs are transferred back to the host in as the run proceeds. There can be a loss of data if the dSPACE board processor is very busy or if there is too much data to transfer back given limits of the connection bandwidth. The advantage of running with the box unchecked is that there are no memory limits, so the model can make very long runs if the hard drive of the host computer has enough space to save the results.

(38) Drop-down control to specify a dSPACE tool option. There are four options:

1. **No dSPACE tool: (ScoutCMD Only)**. Downloading the solver library is controlled by the ScoutCMD program.
2. **ControlDesk + ScoutCMD**: Downloading the solver library will be done by the ScoutCMD program, and ControlDesk will also be launched.
3. **ControlDesk Only**: Launch ControlDesk, which will download the solver library using a Python script.
4. **ControlDesk NG**: Launch ControlDesk NG, which will download the solver library and give run control for the simulation.

(39) Drop-down control for selecting the type of dSPACE board (Figure 36).

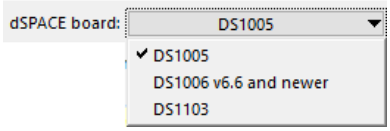


Figure 36, Choose the dSPACE board.

(40) This parameter name informs the Live Animators what computer is running the Live Data Server (i.e. `RundSPACE.exe`). If live animation is used, this parameter must contain the I.P. address or Host Name of the computer running the Live Data Server.