**Transfer Dummy**

**Why are transfer dummies required?**

Monitoring of ram variables related to a signal is possible only if there is ASAP name associated with the variable. Only some of the carmaker / customer specific / user input signals have ASAP names associated with them, hence these signals can be directly monitored in the Datalyser tool by adding the associated ASAP names. The rest of the signals which do not have associated ASAP name’s should be first mapped to transfer dummy signals so that they can be monitored in Datalyser (Transfer dummy signals have ASAP’s associated with them).

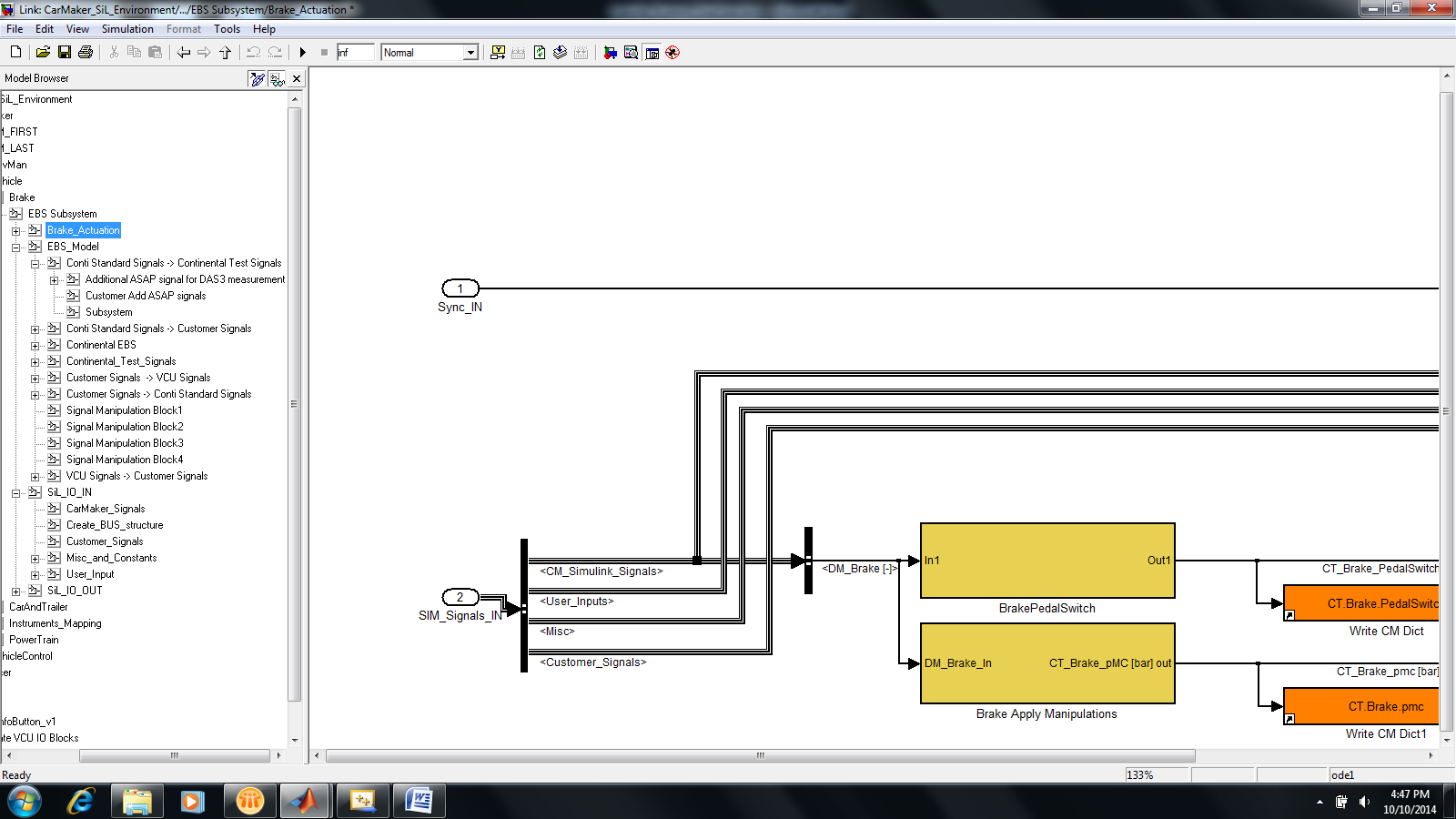
The mapping of the signals to the transfer dummy signals must be done in simulink CarSil environment, the relative path of the library block used for mapping transfer dummy to the above mentioned signals is as follows:

..\..\Project\_Name\Model\_Environment\Vehicle\Brake\EBS\_Model\CT\_standard\_2\_Continental\_Test\_Signals\CT\_standard\_2\_Continental\_Test\_Signals.mdl

The block highlighted in green in screenshot below is **CT\_standard\_2\_Continental\_Test\_Signals** block. Inside this block we have 3 more blocks as show below, the block highlighted in blue color is the location where we need to map the test Signals to the Transfer Dummy signals.

In the above snapshot the section highlighted in grey, the test signals are selected from the input bus selector coming from **SIL\_IO\_IN** block and mapped to the transfer dummy signals which is further linked to Datalyser block

Category of available signals which can be mapped to Transfer Dummy signals



Continental Test Signals block:

The outputs from the block highlighted in green in the above snapshot leads to the block named “Continental\_Test\_Signals”, the final mapping between the selected signals and the transfer dummies are done here. The mapping in this block is already complete and the user need not further modify this block.

On the Datalyser end you can view these Transfer Dummy signals as shown below under the Car maker folder.

Note: If you have associated the required test signal to transfer dummy signal 0 then the value of this signal can be monitored using the ASAP “SIM\_TRANSFER\_DUMMY\_sint32\_00”. The ASAP names associated to transfer dummies can be found in the following file: “..\..\SIM\sim\_add\_ASAP\_vars\_General.h ”

**Why are Virtual Channels required?**

Monitoring and comparing signals in Datalyser tool is always better if they are having the same unit and resolution, in real world situation this may not be always possible. For example, consider a situation where we need to compare the difference between the values coming from **CarMaker** **Steering angle** and the values provided by thebus signal (ECU internal), in most cases the resolution / units between these signals may not match and hence matching them before comparing is better. To facilitate this Datalyser comes with a feature called Virtual Channel which can be used to change one signals unit/resolution to match the reference signal. The steps to be followed for creating a virtual channel explained below:

Click on the icon “Edit Display File” as shown below:

A new window appears. Click on Add Virtual Channel in the middle bottom of the screen as shown below.

A popup is dislplayed, where you can give the name of the virtual channel, the unit which needs to be displayed and the important section i.e. the code section in which you can write the code to calculate return value, for example:

return (get\_data($Signal\_Name$, 0));

Signal name can be any of the ASAP name (This includes transfer dummies).

Whenever we need unit/resolution conversion we need to add the required formula in the related virtual channel code and output the result via return statement. Click on Add to add that virtual channel in the table and the code snippet as shown in the screenshot below:

In the below example we are directly multiplying the value obtained by get\_data function with (180 / 3.14)

return (get\_data($Signal\_Name$, 0) \* 180/3.14);

The coding Syntax of virtual channels is similar to C and hence we have numerous options to handle this data. Below is the example code section of a virtual channel which outputs the difference between 2 signals :

return (get\_data($Signal\_Name\_01$, 0) – get\_data($Signal\_Name\_02$, 0));

\*\*\*\*\*\*\*\*\*\*\*