Appendix A: DIVe Simulink Transfer package

The creation of modules for DIVe is supported by a set of MATLAB functions, which generate the documentation XML files and the s-functions based on simple Simulink models. The functions of this DIVe Simulink Transfer package are mainly identified by the prefix *dst<FunctionName>*. Some of them are also handy to fix/update documentation XMLs of other authoring tools than Simulink.

A.1 Process overview

Module and Simulink s-function generation is supported along the following steps

1. Create the basic Simulink model block with inports and outports according the DIVe Signal List (or its extension processes).
2. Create at least one example data set. Only model data set class types are needed. A standard data set of classType “initIO” is generated during the process by *dstXmlModule*.
3. [Optional] Create support set files e. g.for model configuration and data set loading of non-standard data, if needed.
4. Place the mdl/slx files, the data set files and support set files within the file system structure representing the DIVe logic (according initial SysDM classification logic).
5. Create documentation XML files of data set implementations, support sets and of the module with the MATLAB functions *dstXmlDataSet*, *dstXmlSupportSet* and *dstXmlModule*.
6. Create s-functions of the open Simulink models (creation functions are based on XML information) on a computer with Simulink Coder installation with the functions *buildDIVeSfcn* and/or *fevalAllVersion* of the package.
7. Recreate the module XML with *dstXmlModule* to include the new s-function model sets.
8. [Optional] Add further data set implementations and create the documentation XML with dstXmlDataSet. The parameter indices are then retrieved by the reference data set documentation XML.

A.2 Data set XML generation

The XML of a data set is described in the *DIVeData.xsd*. Among other XML content it defines mandatory attributes for each data set file:

- isStandard: file is a plain parameter m-file or mat-file containing single parameters or vectors.

- executeAtInit: file is executed by the platform during the initialization process.
- copyToRunDirectory: file is copied to the current run directory during the initialization process.

The additional information is queried during the XML creation process for data sets.

The creation of data set XML files is done by dstXmlDataSet, which has the path of the data set folder as argument or is executed in the data set folder as actual MATLAB path.

A.2.1 Prerequisites

- All files of the data set variant are in an own folder with the name of the data set variant.
- [Optional] For complete documentation of the interface with the parameters in the data set files an s-function wrapper mdl/slx is needed for the determination of the parameter indices. The masked wrapper block needs the structure as generated by Simulink Coder context menu function for code generation, which means the s-function block is within another subsystem inside the masked subsystem, hence in 2nd level of the block.

A.2.2 Comfort functionalities

The function dstFileTagCreate for the creation of file attributes contains several comfort functionalties for standard cases in the documentation process. The comfort functions for the data set documentation are described here.

A.2.2.1 isStandard attribute selection

For data set files with the attribute isStandard = ‘1’ is automatically assumed:
executeAtInit = ’0’
copyToRunDirectory = ‘0’

A.3 SupportSet XML generation

The XML of a support set is described in the *DIVeSupportSet.xsd*. Among other XML content it defines mandatory attributes for each support set file:

- executeAtInit: file is executed by the platform during the initialization process.
- copyToRunDirectory: file is copied to the current run directory during the initialization process.

The additional information is queried during the XML creation process for every support set file.

The creation of data set XML files is done by dstXmlSupportSet, which has the path of the support set folder as argument or is executed in the support set folder as actual MATLAB path.

A.4 Module XML generation

The XML of a module is described in the *DIVeModule.xsd*. Among other XML content it defines mandatory attributes for each modelSet file:

- isMain: indication of the main file of the modelSet, containing the basic model block/dll or function
- copyToRunDirectory: file is copied to the current run directory during the initialization process.

The additional information is queried during the XML creation process for every modelSet of the module.

Further the possible data sets and support sets on logical hierarchy levels of species, family and type are analyzed and offered for module inclusion

The creation of data set XML files is done by dstXmlModule, which has the Simulink path to an opened Simulink block of the modules as argument. The Simulink block is needed to determine the inports and outports of the system.

A.4.1 Prerequisites

- At least one Simulink block implementation of the module exists for the determination of ports and their order index. A block implementation must be a Simulink block of BlockType “SubSystem”, but no Stateflow chart (MaskType = ‘Stateflow’) hence the inport and outport names are accessible via block analysis. Masked blocks of BlockType “s-function” with port description in the mask display or stateflow charts can be easily transformed with the *dstSfcnWrapper* and *dstStateflowWrapper* functions.
- The Simulink file of the block implementation resides in a module implementation folder within a file structure, which implements the DIVe classification logic starting from context level, e. g.:
  …\{phys|ctrl|drv|env}\<species>\<family>\<type>\Module\<implementation>\<modelSet>\<SimulinkFile.mdl/slx>
  …\phys\eng\simple\transient\Module\std\open\eng_simple_transiend_std.mdl
- At least one documented variant of each data set class type exists in a folder according the DIVe classification logic in the respective level of the data set, e. g.:
  …\{phys|ctrl|drv|env}\<species>\Data\<classType>\<variant>\<variant.xml>
  …\{phys|ctrl|drv|env}\<species>\<family>\Data\<classType>\<variant>\<variant.xml>
  …\{phys|ctrl|drv|env}\<species>\<family>\<type>\Data\<classType>\<variant>\<variant.xml>
- All required support sets exists in a folder according the DIVe classification logic in the respective level of the data set, e. g.:
  …\{phys|ctrl|drv|env}\<species>\<family>\supportSets\<classType>\<variant>\<

variant.xml>
…\{phys|ctrl|drv|env}\<species>\<family>\<type>\supportSets\<classType>\<variant>\<variant.xml>

- The DIVe signal list is needed during the process to retrieve the extended port documentation information. The signal list can be eiter provided as an Excel file with the Excel sheet *DIVeSignalsPorts* as first sheet or by a MATLAB mat-file that contains the *source* variable, which is generated by reading the specified Excelsheet with the function calls (mat files can be read faster than xlsx):
  source = dbread('DIVe_signals.xlsx',1); save('DIVe_signals.mat','source');
  If one of these files is present on the path structure, where the generated module is placed, it will be automatically be found and loaded. Priority is on folders closer to the selected module Simulink block, hence e. g.
  …\phys\eng\simple\transient\Module\std\open\DIVe_signals.mat
  has priority before
  …\phys\DIVe_signals.xlsx

A.4.2 Comfort functionalities

The function dstFileTagCreate for the creation of file attributes contains several comfort functionalties for standard cases in the documentation process. The comfort functions for the data set documentation are described here.

A.4.2.1 isMain

Model set files are automatically assumed assumed with the attribute isMain = '1' in the case of:

- The file is the only file in this model set
- The file is the mdl file in a model set, in which are only one mdl and one slx file.
- The file is the only mdl or slx file in a model set.

A.4.2.2 copyToRunDirectory

DIVe platform regulation includes adding the path of the used model set to the current MATLAB path, hence the attribute 'copyToRunDirectory' is not needed for standard Simulink models and mex-functions.

- If the model set contains only mdl and slx files the attribute copyToRunDirectory = '0' is set for all files.
- If it is the only mdl or slx file in a model set and the only other file is a mexw32 or mexw64 file the attribute copyToRunDirectory = '0' is set for all files.

A.4.3 Limitations

The DIVe Simulink Transfer package is not suitable or intended to create multiple data set instances (classNames) from a data set class type. Only known usage is so far is the species mec, which is generated from SimulationX.

A.4.4 Wrapper Functions

For convience to provide a DIVe conform Simulink block two function for wrapper generation are available.

A.4.4.1 dstSfcnWrapper

This function creates a Simulink subsystem with named ports from an s-function block, which has port names only as text in the mask display commands. The resulting block of this function is suitable to run *dstXmlModule*. It is not sufficient to run the *dstXmlDataSet* function with determination of parameter indices.

A.4.4.2 dstStateflowWrapper

This function creates a Simulink subsystem with named ports from a Stateflow block. The resulting block of this function is suitable to run *dstXmlModule*. It is not sufficient to run the *dstXmlDataSet* function with determination of parameter indices.

A.5 S-function generation

The complete s-function generation process of DIVe modules originating from simple Simulink models by the Simulink Coder context menu (right click on subsystem) process is coded into one MATLAB function *buildDIVeSfcn*. If the s-functions are needed from several MATLAB versions, the generation process can be automated with the function *fevalAllVersion*.

It is recommended to save the own start commands of the functions in the example calls of the MATLAB functions.

A.5.1 Prerequisites

- A folder structure representing the DIVe classification logic.
- A documentation XML of the module
- At least one variant implementation of each data set class including the documentation XML.
- A Simulink model in the module's model set *open*.
- If only *buildDIVeSfcn* is used, the path of the *DIVe Simulink Transfer package* must be added to the MATLAB path environment.

A.5.2 Comfort functionalities

A.5.2.1 buildDIVeSfcn

- The build function creates a new MATLAB version specific folder for the s-function generation *create_ <bitSystem>_<MATLABversion>* inside the folder of the open model set and copies the Simulink model to this folder. This enables simultaneous creation of s-functions by several versions.

- The build functions load all reference data sets specified in the module documentation XML.
- The build function renames all s-function parameters to a simple MATLAB workspace variable (needed for correct parameter determination of Simulink Coder).
- The build function transfers all loaded parameters from the data sets to corresponding simple MATLAB workspace variables.
- The build function sets all s-function parameters to tunable parameters for the Simulink Coder.
- The build function invokes the build process. If an error occurs, the error message structure is saved in *xErrorBuild.mat* in the create folder and the create folder is not deleted.
- The generated s-function and its wrapper Simulink model are saved/copied to the respective model set folder (e. g. sfcn_w32_R2010bSP1, sfcn_w64_R2013b). The create folder is deleted after the copy action.
- A test model is build with inputs of data set initIO values and simulated. If the simulation ends without error, the test model is deleted.
- If an error occurs, the test model is saved as <modelName>test.mdl/slx and the error message structure is saved as xErrorRun.mat in the s-function model set folder. To re-run the test open the test model and load the model data via dmdLoadData(<pathModuleXml>).

A.5.2.2 fevalAllVersion

- The function can add specified paths to the MATLAB path.
- The function evaluates all installed MATLAB versions via system path environment settings.
- The user can select the desired MATLAB versions for function evaluation.

A.5.3 Limitations

The *buildDIVeSfcn* works only correctly with modules, which have solely data set classes with the attribute isStandard = ‘1’. For other modules the code has to be adapted for data set loading. The complete data is needed for s-function generation as the Simulink Coder performs a Simulink initialization command, which checks all parameter availabilities.

Only Simulink blocks, which work with the Simulink Coder context menu function for s-function generation of SubSystem blocks, can be used with the *buildDIVeSfcn*.

The s-function generation in R2010bSP1 64bit (and all other service pack levels) requires a compiler setting, which is different from the R2010bSP1 32bit setting. This setting can be only changed interactively via *mex –setup* in the manually opened MATLAB instance. Hence it is currently not possible to generate s-functions by *fevalAllVersion* in R2010bSP1 32bit and R2010bSP1 64bit in the same run.

S-function generation of Simulink blocks saved in R2013b fails in R2013a due to changed model/block properties and object parameters.