Simulink Model A2L File Generation Manual

Release 2021-A – May 2021

dSPACE

How to Contact dSPACE

Mail:	dSPACE GmbH Rathenaustraße 26 33102 Paderborn Germany
Tel.:	+49 5251 1638-0
Fax:	+49 5251 16198-0
E-mail:	info@dspace.de
Web:	http://www.dspace.com

How to Contact dSPACE Support

If you encounter a problem when using dSPACE products, contact your local dSPACE representative:

- Local dSPACE companies and distributors: http://www.dspace.com/go/locations
- For countries not listed, contact dSPACE GmbH in Paderborn, Germany.
  Tel.: +49 5251 1638-941 or e-mail: support@dspace.de

You can also use the support request form: http://www.dspace.com/go/supportrequest. If you are logged on to mydSPACE, you are automatically identified and do not need to add your contact details manually.

If possible, always provide the relevant dSPACE License ID or the serial number of the CmContainer in your support request.

Software Updates and Patches

dSPACE strongly recommends that you download and install the most recent patches for your current dSPACE installation. Visit http://www.dspace.com/go/patches for software updates and patches.

Important Notice

Contents

About This Document

Content

This document provides access to information you need to generate A2L files for your specific dSPACE simulation platform according to the ASAM MCD-2 MC standard.

Symbols

dSPACE user documentation uses the following symbols:

Symbol	Description
DANGER	Indicates a hazardous situation that, if not avoided, will result in death or serious injury.
WARNING	Indicates a hazardous situation that, if not avoided, could result in death or serious injury.
CAUTION	Indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.
NOTICE	Indicates a hazard that, if not avoided, could result in property damage.
Note	Indicates important information that you should take into account to avoid malfunctions.
Tip	Indicates tips that can make your work easier.
	Indicates a link that refers to a definition in the glossary, which you can find at the end of the document unless stated otherwise.
	Precedes the document title in a link that refers to another document.

Naming conventions

dSPACE user documentation uses the following naming conventions:

%name% Names enclosed in percent signs refer to environment variables for file and path names.

< > Angle brackets contain wildcard characters or placeholders for variable file and path names, etc.

Special folders

Some software products use the following special folders:

Common Program Data folder A standard folder for application-specific configuration data that is used by all users.

`%PROGRAMDATA%\dSPACE\<InstallationGUID>\<ProductName>`

or

`%PROGRAMDATA%\dSPACE\<ProductName>\<VersionNumber>`

Documents folder A standard folder for user-specific documents.

`%USERPROFILE%\Documents\dSPACE\<ProductName>\<VersionNumber>`

Local Program Data folder A standard folder for application-specific configuration data that is used by the current, non-roaming user.

`%USERPROFILE%\AppData\Local\dSPACE\<InstallationGUID>\<ProductName>`

Accessing dSPACE Help and PDF Files

After you install and decrypt dSPACE software, the documentation for the installed products is available in dSPACE Help and as PDF files.

dSPACE Help (local) You can open your local installation of dSPACE Help:

- On its home page via Windows Start Menu
- On specific content using context-sensitive help via **F1**

dSPACE Help (Web) You can access the Web version of dSPACE Help at www.dspace.com/go/help.

To access the Web version, you must have a *mydSPACE* account.

PDF files You can access PDF files via the  icon in dSPACE Help. The PDF opens on the first page.

Introduction and Overview

Where to go from here

Information in this section

Basics on A2L File Generation Based on Simulink® Coder™

Introduction

Simulink® Coder™ supports A2L file generation in connection with model code generation. dSPACE-specific extensions to the Simulink® Coder™-based A2L file generation let you create A2L files for your specific dSPACE simulation platform according to the ASAM MCD-2 MC standard. Such A2L files list the variables of the application that is generated for that platform and make them available to measurement and calibration systems.

dSPACE products supporting A2L file generation

The following dSPACE products support A2L file generation based on Simulink® Coder™:

dSPACE Product	System Target File
RTI Bypass - Internal Bypassing Option	`rti_intbyp_<target type>`[1]`.tlc`
RTI for DS1007-based systems	`rti1007.tlc`
RTI for MicroAutoBox II	`rti1401.tlc`
RTI for MicroLabBox	`rti1202.tlc`
Model Interface Package for Simulink	`dsrt.tlc` **Note** If you use the `dsrt.tlc` system target file, the generated A2L file fragment is not yet ready to use. It is finalized during the build process.

[1] `<target type>` stands for the microcontroller type that the internal bypass functions are to be generated for.

Supported MATLAB Releases For information on the supported MATLAB Releases, refer to Required MATLAB Releases (Installing dSPACE Software 🕮).

A2L file contents

The generated A2L file contains descriptions such as the memory addresses and data types of measurement and calibration objects. In the A2L file:

- Measurement variables can be recognized by the `MEASUREMENT` A2L keyword. A2L file generation adds `Simulink.Signal` objects to an A2L file as `MEASUREMENT` entries.
- Calibration variables – either scalars, or 1- or 2-dimensional look-up tables – can be recognized by the `CHARACTERISTIC` A2L keyword. A2L file generation adds `Simulink.Parameter` objects to an A2L file as `CHARACTERISTIC` entries.

The A2L file also includes interface-specific information (`IF_DATA`) used to access the dSPACE simulation platform.

A2L file generation features

A2L file generation provides the following features:

- Support of Simulink model referencing, including referenced protected models.

  One A2L file is generated for each Simulink top model (= topmost model in a hierarchy of referenced models). For DS1007-based multiprocessor systems, one A2L file is generated for each multiprocessor submodel.
- Generation of A2L files according to ASAM MCD-2 MC Version 1.6.1

- Functions for adding signals and parameters of the Simulink model to the A2L file automatically. Refer to Adding Parameters and Signals to the A2L File on page 11.
- Insertion of calibration protocol- and interface-specific `IF_DATA` entries
- Insertion of simulation platform-specific entries to the A2L file, such as memory segment information
- Support of Simulink fixed-point data types
- Support of Simulink enumerations
- Generation of an EPK identifier for consistency checks (supported by RTI for MicroAutoBox II only)

  When you build the real-time application for MicroAutoBox II, an EPK identifier is automatically included in the A2L file and in the real-time application.

  The EPK identifier is different for each application. It can be used by an measurement and calibration system to check whether the A2L file is consistent with the currently running real-time application. If you use ControlDesk, the check is performed when online calibration is started.

  In the A2L file, the EPK identifier is specified by the `ADDR_EPK` and `EPK` attributes of the `MOD_PAR` element.

Configuring A2L file generation

You can configure the A2L file to be generated by a dSPACE-specific MATLAB workspace configuration structure. Refer to Basics on Configuring A2L File Generation on page 18.

Limitations

A2L file generation based on Simulink® Coder™ has some limitations. Refer to Limitations for A2L File Generation on page 36.

Migrating to A2L File Generation of dSPACE Release 2016-B and Later

Introduction

You might have to carry out some migration steps when you migrate to A2L file generation of dSPACE Release 2016-B and later.

Incorrect A2L file entries for specific blocks (dSPACE Release 2016-A and earlier)

In some cases, A2L files generated with dSPACE products from dSPACE Release 2016-A and earlier contained incorrect entries.

Incorrect A2L file entries were generated for the following blocks:

- Interpolation using PreLookup blocks
- Constant blocks referencing a `Simulink.Parameter` object with matrix values

In the generated A2L file, the rows and columns of the matrices and look-up tables generated for these blocks were transposed erroneously. As a result, the representation of the matrices and look-up tables in a measurement

and calibration tool such as dSPACE ControlDesk was not consistent with their representation in Simulink.

Corrected A2L file generation with dSPACE Release 2016-B and later

A2L file generation is correct with dSPACE products from dSPACE Release 2016-B and later. As a result, the representation of the matrices and look-up tables in a measurement and calibration tool such as ControlDesk is consistent with their representation in Simulink.

Migration steps

When you reload a corrected A2L file in a measurement and calibration tool such as ControlDesk, you have to adapt layouts and data sets based on the faulty A2L file entries accordingly.

Changed A2L File Generation for the dSPACE Run-Time Target as of dSPACE Release 2019-A

Modified A2L file generation

With dSPACE Release 2019-A, the generation of A2L files for the dSPACE Run-Time Target (`dsrt.tlc`) has changed: The **Configuration Parameters** dialog now provides the **Variable description file format** property, which lets you set A2L file generation for the dSPACE Run-Time Target. For more information, refer to How to Generate an A2L File on page 29.

Related topics

HowTos

How to Generate an A2L File....29

Adding Parameters and Signals to the A2L File

Where to go from here

Information in this section

Adding Parameters and Signals to the A2L File

Introduction

Parameters and signals of a Simulink model appear in the A2L file if:

- They are defined as `Simulink.Parameter` and `Simulink.Signal` objects in the MATLAB workspace.
- Their `StorageClass` is set to `ExportedGlobal`.

Generating Simulink.Parameter and Simulink.Signal objects and setting the StorageClass

The following API functions let you create `Simulink.Parameter` objects for model parameters and `Simulink.Signal` objects for model signals in the MATLAB workspace. In addition, the `StorageClass` of these objects is set to `ExportedGlobal`. As a result, the model parameters and signals are configured

by the API functions so that they will be included in the A2L file generation process.

- `ds_asap2paramcreate` (for generating `Simulink.Parameter` objects for all the parameters of a Simulink model based on workspace variables)
- `ds_asap2signalcreate` (for generating `Simulink.Signal` objects for all the signals of a Simulink model)

Call these functions manually when you use one of the dSPACE products that support A2L file generation based on Simulink® Coder™.

When you use the RTI Bypass Blockset (Internal Bypassing Option), the `ds_asap2paramcreate` and the `ds_asap2signalcreate` functions are called when you enable the **Generate Simulink objects from parameters** and the **Generate Simulink objects from signal names** options on the Build Page (RTIBYPASS_SETUP_BLx for INTERNAL) (RTI Bypass Blockset Reference 🕮).

Clearing Simulink.Parameter and Simulink.Signal objects

You can clear the `Simulink.Parameter` and `Simulink.Signal` objects from the MATLAB workspace to avoid parameters and/or signals from being generated into the A2L file. Refer to Clearing Objects Created Automatically on page 14.

Related topics

Basics

Clearing Objects Created Automatically....14

HowTos

How to Generate an A2L File....29

References

ds_asap2paramcreate....12
ds_asap2signalcreate....13

ds_asap2paramcreate

Purpose

To prepare workspace variables used in a Simulink model for being included in the A2L file generation process.

Syntax

```
paramObjList = ds_asap2paramcreate(model)
```

Description

The function generates `Simulink.Parameter` objects for all numeric referenced workspace variables. In addition, the `StorageClass` of these objects is set to `ExportedGlobal`.

Parameter

The following parameter is available:

Parameter	Description
`model`	Name or handle of the Simulink model

Return parameter

The following return parameter is available:

Parameter	Description
`paramObjList`	List of the converted variable names

Related topics

Basics

Adding Parameters and Signals to the A2L File..... 11

ds_asap2signalcreate

Purpose

To prepare labels used in a Simulink model for being included in the A2L file generation process.

Syntax

```
signalObjList = ds_asap2signalcreate(model)
```

Description

The function generates a `Simulink.Signal` object for each label in the model. In addition, the `StorageClass` of such an object is set to `ExportedGlobal`.

The following preconditions apply:

- Signals must be labeled with a valid workspace variable name.
- The signal name:
  - Must be unique within the entire Simulink model
  - Must not contain more than 63 characters

Parameter

The following parameter is available:

Parameter	Description
`model`	Name or handle of the Simulink model

Return parameter

The following return parameter is available:

Parameter	Description
`signalObjList`	List of the generated signal names

Related topics

Basics

Adding Parameters and Signals to the A2L File 11

Clearing Objects Created Automatically

Introduction

You can:

- Clear all the `Simulink.Signal` objects from the MATLAB workspace.
- Restore the original workspace variables.

Clearing Simulink.Signal objects

To clear all the `Simulink.Signal` objects (created via the `ds_asap2signalcreate.m` function) from the MATLAB workspace, run the `<modelname>_asap2clear` script.

All the `Simulink.Signal` objects created automatically are deleted. This affects no other variables.

This avoids the objects from being generated into the A2L file in a subsequent build process.

Restoring the original workspace variables

To restore the original workspace variables that were in the workspace *before* the `ds_asap2paramcreate` function was called, run the created `<modelname>_ds_ASAP2VarsBackup` MAT file.

The created `Simulink.Parameter` objects are overwritten.

This avoids the objects from being generated into the A2L file in a subsequent build process.

Related topics

Basics

Adding Parameters and Signals to the A2L File..... 11

Configuring A2L File Generation

Where to go from here

Information in this section

Basics on Configuring A2L File Generation

dSPACE-specific MATLAB configuration structure

A2L file generation based on Simulink® Coder™ lets you use a dSPACE-specific MATLAB workspace configuration structure, which allows you to alter A2L file generation. You can use the structure to add or modify property information for calibration and measurement variables, function-based grouping of variables, and computation methods to the A2L file.

Providing Additional Information for ECU Variables

Introduction

You can use the dSPACE-specific MATLAB workspace configuration structure to provide additional information for an ECU variable.

Structure with properties for variable generation

The following MATLAB configuration structure elements can be used to provide additional information for an ECU variable.

The structure has the following general form:

```
DSPACE_Config.<ModelName>.VARIABLES.<ObjectName>.PROPERTIES
    DisplayID: string
    LongID: string
    MinVal: double
    MaxVal: double
    Format: string
    CompuMethod: string
    WorkPt: string
    BitMask: string
    RASTER: [1x1 struct]
    AXIS: [1x1 struct]
```

- `<ModelName>` is a placeholder for the name of your model.
- `<ObjectName>` is a placeholder for the name of the Simulink object.

Property	Description
`DisplayID`	Applies to the `DISPLAY_IDENTIFIER` A2L parameter of the variable. It can be used to specify a display name.
`LongID`	Applies to the `LongIdentifier` A2L parameter of the variable. It can be used to specify a comment or description for the ECU variable.
`MinVal`	Applies to the `LowerLimit` A2L parameter of the variable. It can be used to specify a lower limit of the value range for the ECU variable.
`MaxVal`	Applies to the `UpperLimit` A2L parameter of the variable. It can be used to specify an upper limit of the value range for the ECU variable.
`Format`	Applies to the `FORMAT` A2L parameter of the variable. It can be used to specify a special display format for the ECU variable.

Property	Description
CompuMethod	Applies to the Conversion A2L parameter of the variable. It can be used to specify a reference to the computation method used with the ECU variable.
WorkPt	For look-up tables (LUT) only: An entry with the COMPARISON_QUANTITY keyword is added to the CHARACTERISTIC entry if the corresponding MEASUREMENT entry exists in the A2L file.
BitMask	Defines a bit mask using the BIT_MASK keyword.
RASTER	Applies to the Daq_Event element in the IF_DATA XCP section of the MEASUREMENT or CHARACTERISTIC block in the AML definition of the XCP protocol layer. The Daq_Event element can be used to specify measurement raster information to the ECU variable. See below. If measurement raster information is available, an IF_DATA XCP section is added to the MEASUREMENT or CHARACTERISTIC block, relating the ECU variable to the measurement raster.
AXIS	For look-up tables (LUT) only: Substructure containing properties of the related axes (optional). See below.

Example: Generating a display identifier for a variable The following example shows how to generate a display ID for the Simulink object Const1 for use in the demoRTIBypassINTERNALModel model:

```
>> DSPACE_Config.demoRTIBypassINTERNALModel.VARIABLES.Const1.
PROPERTIES.DisplayID = 'myDisplayID'
```

The resulting A2L file entry is:

```
/begin CHARACTERISTIC
  Const1
  ...
  DISPLAY_IDENTIFIER myDisplayID
/end CHARACTERISTIC
```

Structure with properties for the measurement raster definition

The following MATLAB configuration structure elements can be used to generate a measurement raster definition.

The structure has the following general form:

- For a fixed raster reference:

```
DSPACE_Config.<ModelName>.VARIABLES.<ObjectName>.
PROPERTIES.RASTER
    Fixed: {uint, uint, ...}
```

- For a variable raster reference:

```
DSPACE_Config.<ModelName>.VARIABLES.<ObjectName>.
PROPERTIES.RASTER
    Default: {uint}
    Available: {uint, uint, ...}
```

- <ModelName> is a placeholder for the name of your model.
- <ObjectName> is a placeholder for the name of the Simulink object.

Property	Description
`Fixed`	Applies to the `FIXED_EVENT_LIST` taggedstruct definition in the `Daq_Event` section in the MEASUREMENT or CHARACTERISTIC block. It can be used to specify a list of measurement rasters (list of service IDs).
`Default`	Applies to the `DEFAULT_EVENT_LIST` taggedstruct definition of the optional `VARIABLE` element in the `Daq_Event` section in the MEASUREMENT or CHARACTERISTIC block. It can be used to specify a default measurement raster (list with a service ID).
`Available`	Applies to the `AVAILABLE_EVENT_LIST` taggedstruct definition of the optional `VARIABLE` element in the `Daq_Event` section in the MEASUREMENT or CHARACTERISTIC block. It can be used to specify a list of available measurement rasters (list with service IDs).

Example: Defining a fixed raster reference The following example shows how to define a fixed raster reference in the A2L file for the Simulink object Counter1 for use in the demoRTIBypassINTERNALModel model:

```
>> DSPACE_ConfigdemoRTIBypassINTERNALModel.VARIABLES.
Counter1.PROPERTIES.RASTER.Fixed = {0, 1}
```

The resulting A2L file entry is:

```
/begin MEASUREMENT
  Counter1
  ...
  /begin IF_DATA XCP
    /begin DAQ_EVENT
      /begin FIXED_EVENT_LIST
        0
        1
      /end FIXED_EVENT_LIST
    /end DAQ_EVENT
  /end IF_DATA
/end MEASUREMENT
```

Example: Defining a variable raster reference The following example shows how to define a variable raster reference in the A2L file for the Simulink object Counter2 for use in the demoRTIBypassINTERNALModel model:

```
>> DSPACE_ConfigdemoRTIBypassINTERNALModel.VARIABLES.
Counter2.PROPERTIES.RASTER.Default = {3}
>> DSPACE_ConfigdemoRTIBypassINTERNALModel.VARIABLES.
Counter2.PROPERTIES.RASTER.Available = {2, 3, 10}
```

The resulting A2L file entry is:

```
/begin MEASUREMENT
  Counter2
  ...
  /begin IF_DATA XCP
    /begin DAQ_EVENT
      VARIABLE
      /begin DEFAULT_EVENT_LIST
        3
      /end DEFAULT_EVENT_LIST
      /begin AVAILABLE_EVENT_LIST
        2
        3
        10
      /end AVAILABLE_EVENT_LIST
    /end DAQ_EVENT
  /end IF_DATA
/end MEASUREMENT
```

Structure with properties for look-up table generation

The following MATLAB configuration structure elements can be used to configure look-up table generation.

The structure has the following general form:

```
DSPACE_Config.<ModelName>.VARIABLES.<ObjectName>.
PROPERTIES.AXIS
    Transpose: bool
    ForceShared: bool
```

- `<ModelName>` is a placeholder for the name of your model.
- `<ObjectName>` is a placeholder for the name of the Simulink object.

You can use the `Transpose` and `ForceShared` entries independent of each other. They are switched on for values greater than 0 and off otherwise.

Property	Description
`Transpose`	To transpose, i.e., interchange the X- and Y-axes of a 2-D look-up table: ▪ 0: Axes are not transposed ▪ 1: Axes are transposed (compared with Simulink Coder default)
`ForceShared`	To force the generation of shared axes (`COM_AXIS` entries) instead of embedded axes (`STD_AXIS` entries) for 1- and 2-D look-up tables: ▪ 0: Generation of `STD_AXIS` entries ▪ 1: Generation of `COM_AXIS` entries

Example: Defining the representation of the X- and Y-axes The following example shows how to define the representation of the x- and y-axes in the A2L file for the `lut2d_value` Simulink object for use in the demo_model model. In this example, the x-axis and y-axis are swapped.

```
>> DSPACE_Config.demo_model.VARIABLES.lut2d_value.PROPERTIES.
AXIS.Transpose = 1
```

The resulting A2L file entry is:

```
/begin CHARACTERISTIC
  /* Name                */    lut2d_value
  /* Long identifier     */    ""
  /* Characteristic Type */    MAP
  /* ECU Address         */    0x002BBF88
  /* Record Layout       */    Lookup2D_FLOAT64_IEEE_TRANSPOSED
  /* Maxdiff             */    0
  /* Conversion method   */    demo_model_CM_double
  /* Lower limit         */    -1.7E+308
  /* Upper limit         */    1.7E+308
  /begin AXIS_DESCR
    /* Description of X-Axis Points */
    /* Axis Type          */    COM_AXIS
    /* Reference to Input */    NO_INPUT_QUANTITY
    /* Conversion method  */    demo_model_CM_double
    /* Number of Axis Pts */    4
    /* Lower limit        */    -1.7E+308
    /* Upper limit        */    1.7E+308
    AXIS_PTS_REF                lut2d_y_axis2
  /end AXIS_DESCR
  /begin AXIS_DESCR
    /* Description of Y-Axis Points */
    /* Axis Type          */    COM_AXIS
    /* Reference to Input  */     NO_INPUT_QUANTITY
    /* Conversion method  */    demo_model_CM_double
    /* Number of Axis Pts */    3
    /* Lower limit        */    -1.7E+308
    /* Upper limit        */    1.7E+308
    AXIS_PTS_REF                lut2d_x_axis1
  /end AXIS_DESCR
/end CHARACTERISTIC
```

Related topics

References

Basics on Configuring A2L File Generation.....18

Structuring Parameters and Signals of ECU Functions

Introduction

You can use the dSPACE-specific MATLAB workspace configuration structure to generate information on the hierarchical grouping of parameters and signals.

Structure with properties for grouping parameters and signals in any hierarchy

The following MATLAB configuration structure elements can be used to generate information on the hierarchical grouping of parameters and signals.

The structure has the following general form:

```
DSPACE_Config.<ModelName>.FUNCTIONS.<FunctionName>.
PROPERTIES
    FUNCTIONS:
    IN_MEASUREMENT:
    OUT_MEASUREMENT:
    LOC_MEASUREMENT:
    DEF_CHARACTERISTIC:
    REF_CHARACTERISTIC:
```

- `<ModelName>` is a placeholder for the name of your model.
- `<FunctionName>` is a placeholder for the name of the function. The function has the following structure:

  ```
  /begin FUNCTION <FunctionName>
     <function body>
  /end FUNCTION
  ```

Property	Description
`FUNCTIONS`	Recursive definitions of subfunctions Each subfunction also has the structure `<FunctionName>.PROPERTIES`, where the `PROPERTIES` element is optional.
`IN_MEASUREMENT`	Reference to MEASUREMENT object that is defined as input to the function
`OUT_MEASUREMENT`	Reference to MEASUREMENT object that is defined as output of the function
`LOC_MEASUREMENT`	Reference to MEASUREMENT object that is defined as a local variable in the function
`DEF_CHARACTERISTIC`	Reference to CHARACTERISTIC object that belongs to the function
`REF_CHARACTERISTIC`	Reference to CHARACTERISTIC object that is used but not owned by the function

Note

When you generate an A2L file for a model containing referenced models, no structure information based on the dSPACE-specific MATLAB configuration structure via the `FUNCTION` keyword is generated.

Example: Generating an input variable for a function The following example shows how to generate Const1 as an input variable for the myfun function for use in the demoRTIBypassINTERNALModel model:

```
>> DSPACE_Config.demoRTIBypassINTERNALModel.FUNCTIONS.myfun.
    PROPERTIES.IN_MEASUREMENT = {'Const1'}
```

The resulting A2L file entry is:

```
/begin FUNCTION
  myfun
  ...
  /begin IN_MEASUREMENT
    Const1
  /end IN_MEASUREMENT
/end FUNCTION
```

Related topics

References

Basics on Configuring A2L File Generation 18

Providing Additional Information for Computation Method Generation

Introduction

You can use the dSPACE-specific MATLAB workspace configuration structure to provide additional information for computation method generation.

Structure with properties for computation method generation

The following MATLAB configuration structure elements can be used to provide additional information for computation method generation.

The structure has the following general form:

```
DSPACE_Config.<ModelName>.COMPUTATIONS.<MethodName>.
PROPERTIES
    Type: string
    LongID: string
    Format: string
    Unit: string
    Pairs: [1x2 struct]
```

- `<ModelName>` is a placeholder for the name of your model.
- `<MethodName>` is a placeholder for the name of the computation method.

Property	Description
`Type`	Applies to the `ConversionType` A2L parameter. Currently, only the `TAB_VERB` (verbal conversion table) type is supported.
`LongID`	Applies to the `LongIdentifier` A2L parameter. It can be used to specify a comment or description for the computation method.
`Format`	Format string (for example, `"%15.10"`)
`Unit`	Physical unit for the converted value
`Pairs`[1)]	Applies to the `NumberValuePairs` A2L parameter. This parameter is used to describe a number of successive value/name pairs where `Name` is a string and `Value` a numerical value.

1) Pairs are optional. If not available, only the reference to another COMPU_VTAB is generated.

Example: Specifying the output format and unit for converted values, and generating value/name pairs The following example shows how to specify the output format and physical unit for the converted values, and how to generate value/name pairs for the computation method COMPU_METHOD_1 for use in the demoRTIBypassINTERNALModel model:

```
>> DSPACE_Config.demoRTIBypassINTERNALModel.COMPUTATIONS.COMPU_METHOD_1.
PROPERTIES.Type = 'TAB_VERB'
>> DSPACE_Config.demoRTIBypassINTERNALModel.COMPUTATIONS.COMPU_METHOD_1.
PROPERTIES.Format = '%15.10'
>> DSPACE_Config.demoRTIBypassINTERNALModel.COMPUTATIONS.COMPU_METHOD_1.
PROPERTIES.Unit = 'cm'
```

```
>> DSPACE_Config.demoRTIBypassINTERNALModel.COMPUTATIONS.COMPU_METHOD_1.
PROPERTIES.Pairs(1) = struct('Name','On','Value',1)
>> DSPACE_Config.demoRTIBypassINTERNALModel.COMPUTATIONS.COMPU_METHOD_1.
PROPERTIES.Pairs(2) = struct('Name','Off','Value',0)
```

The resulting A2L file entry is similar to the following:

```
/begin COMPU_METHOD
  COMPU_METHOD_1
  TAB_VERB
  "%15.10"
  "°C"
  COMPU_TAB_REF CompuVtab_01
/end COMPU_METHOD

/begin COMPU_VTAB
  CompuVtab_01
  ...
  TAB_VERB
  2
  1 "On"
  0 "Off"
/end COMPU_VTAB
```

Related topics

References

Basics on Configuring A2L File Generation 18

Adding User-Defined Information

Introduction

You can use the dSPACE-specific MATLAB workspace configuration structure to add user-defined information.

Structure for user-defined information generation

The following MATLAB configuration structure elements can be used to add user-defined information to the `MOD_PAR` block of the A2L file.

The structure has the following general form:

```
DSPACE_Config.<ModelName>.USER_INFO
    Supplier: string
    Customer: string
    User: string
    Phone_No: string
```

`<ModelName>` is a placeholder for the name of your model.

All fields are optional.

Property	Description
`Supplier`	Supplier name
`Customer`	Customer name
`User`	User name
`Phone_No`	Phone number

Related topics

References

Basics on Configuring A2L File Generation..... 18

Specifying the Layout of Two-Dimensional CHARACTERISTIC and MEASUREMENT Variables

Introduction

You can use the dSPACE-specific MATLAB workspace configuration structure to specify the layout of two-dimensional `CHARACTERISTIC` and `MEASUREMENT` variables.

Structure for LAYOUT specification

You can use the MATLAB configuration structure to switch the layout of two-dimensional `CHARACTERISTIC` and `MEASUREMENT` variables from the `ROW_DIR` default layout type to the `COLUMN_DIR` layout type. This changes the representation of these A2L file entries in measurement and calibration tools, such as ControlDesk.

To switch the layout type to `COLUMN_DIR`, set the `DS_CUSTOM_LAYOUT` field to `COLUMN`:

```
DSPACE_Config.DS_CUSTOM_LAYOUT= 'COLUMN'
```

In any other case, the `ROW_DIR` default layout type is used for A2L file generation.

Note

The `DS_CUSTOM_LAYOUT` entry is specified on the highest level of the dSPACE-specific MATLAB workspace configuration structure, i.e., *it is applied to A2L file generation of all models*. You cannot specify the layout in a model-specific way.
However, you can transpose, i.e., interchange the X- and Y-axes specifically for 2-D look-up tables. Refer to Structure with properties for look-up table generation on page 21.

Related topics

References

Basics on Configuring A2L File Generation..... 18

Generating A2L Files for Simulink Models

Where to go from here

Information in this section

How to Generate an A2L File

Objective

Generating an A2L file using Simulink® Coder™ lets you list the variables located in the memory of the dSPACE simulation platform and make them available to measurement and calibration systems.

Preconditions

- Parameters and signals of a Simulink model appear in the A2L file if:
  - They are defined as `Simulink.Parameter` and `Simulink.Signal` objects in the MATLAB workspace.
  - Their `StorageClass` is set to `ExportedGlobal`.

Tip

To let parameters and signals of a Simulink model appear in the A2L file, you can use the `ds_asap2paramcreate` and the `ds_asap2signalcreate` functions. See Adding Parameters and Signals to the A2L File on page 11 for function details.

- To let signals appear in the A2L file, you also have to ensure that one of the following conditions is fulfilled:
  - Either the **Signal must resolve to Simulink.Signal object** option is selected in the signal's **Signal Properties** dialog.
  - Or the **Signal resolution** option is set to **Explicit and implicit** on the **Data Validity** page of the **Diagnostics** dialog.

Restrictions

For signals to appear in the A2L file, the following restriction applies:

Virtual signals, such as the output signals of Mux blocks, are not stored in consecutive memory locations and cannot be accessed as an array of variables. For this reason, output signals of virtual blocks cannot be added to the A2L file.

Tip

To ensure that all parts of a vectorized signal are stored in consecutive memory locations, use a Signal Conversion block with the **Output** parameter set to "Signal copy" and label the Signal Conversion block output instead of the output of the virtual block.

Possible methods

- Method to generate an A2L file for the dSPACE Run-Time Target or for an RTI target: Refer to Method 1.
- RTI Bypass Blockset (Internal Bypassing Option)-specific method: Refer to Method 2.

Method 1

To generate an A2L file (dSPACE Run-Time Target or RTI target)

1. In the model, press **Ctrl** + **E** to open the **Configuration Parameters** dialog.
2. On the **Code Generation** page, specify the system target file that corresponds to your simulation platform.

   For example, specify `dsrt.tlc` as the system target file for the *dSPACE Run-Time Target*.
3. *For the dSPACE Run-Time Target:*

   On the **Code Generation - DSRT variable description file options** page, select **A2L** from the **Variable description file format** list.

   For an RTI target:

On the **Code Generation - RTI variable description file options** page, select **TRC and A2L** from the **Variable description file format** list.

Note

In model referencing hierarchies, all models must have the same configuration.

4 Click **OK** to apply your settings and close the dialog.

5 Start the build process.

- For the dSPACE Run-Time Target, code is generated, and a Simulink implementation container (SIC) file containing an A2L file fragment is created.

  In a next step, import the SIC file to VEOS or to ConfigurationDesk. During the build process, the A2L file fragment is extended by simulation platform-specific information such as memory addresses during the build process.

  Refer to:

  - How to Import Simulink Implementations (VEOS Manual 🕮)
  - Adding Simulink Implementation Containers to a ConfigurationDesk Application (ConfigurationDesk Real-Time Implementation Guide 🕮)
- For an RTI target, the build process is started. For more information on the build process, refer to Building and Downloading the Model (RTI and RTI-MP Implementation Guide 🕮).

Method 2

To generate an A2L file (RTI Bypass Blockset (Internal Bypassing Option) only)

1 Select the following options on the **Build** page of the RTIBYPASS_SETUP_BLx (INTERNAL) block:

- **Generate extended A2L**
- **Generate Simulink Objects from Signal Names**
- **Generate Simulink Objects from Parameters**

Refer to Build Page (RTIBYPASS_SETUP_BLx for INTERNAL) (RTI Bypass Blockset Reference 🕮).

2 Start the build process.

The build process is started. For more information on the build process, refer to Building and Downloading the Model (RTI and RTI-MP Implementation Guide 🕮).

Result

The A2L file is generated.

Related topics

Basics

Adding Parameters and Signals to the A2L File 11

References

Details on the Generated A2L File 32

Details on the Generated A2L File

A2L file frame

The A2L file is framed by a `PROJECT` block (named after the Simulink model name), which contains a `HEADER` block and exactly one `MODULE` block (named after the Simulink model name). The frame is generated by the Simulink® Coder™.

MODULE block

All settings specified in the ASAP2 metalanguage are placed at the beginning of the `MODULE` block.

MOD_PAR block

The following `MOD_PAR` block specifies user-specific data like the name and the phone number. It also contains the required `MEMORY_SEGMENT`.

MOD_COMMON block

The `MOD_COMMON` block is standard for all the supported dSPACE platforms.

RECORD_LAYOUT blocks

`RECORD_LAYOUT` blocks are generated into the A2L file even if no parameters are specified. Record layouts are referenced by parameters and table axes only. There are different record layouts for scalar variables, and 1-D and 2-D look-up tables. A `CHARACTERISTIC` block is written to the A2L file for each parameter.

COMPU_METHOD blocks

The `COMPU_METHOD`s are specified at the end of the `MODULE` block. Two measurement or characteristic variables use the same conversion method if the physical units are identical. The conversion methods for different variable types are identical if their physical units match.

CHARACTERISTIC and MEASUREMENT entries

CHARACTERISTIC entries `CHARACTERISTIC` entries and related entries such as `COMPU_METHOD`s are added to the A2L file for `Simulink.Parameter` objects whose `StorageClass` is set to `ExportedGlobal`.

MEASUREMENT entries `MEASUREMENT` entries and related entries are added to the A2L file for `Simulink.Signal` objects whose `StorageClass` is set to `ExportedGlobal`.

Configuring CHARACTERISTIC and MEASUREMENT entries The following table lists the properties of `Simulink.Parameter` and `Simulink.Signal` objects that influence A2L file generation.

Keep in mind that you can influence A2L file generation also via a dSPACE-specific MATLAB configuration structure. See Basics on Configuring A2L File Generation on page 18.

Property of Simulink Objects	Effect on CHARACTERISTIC / MEASUREMENT A2L File Entries
`Alias`	If the `Alias` property of a `Simulink.Parameter` or a `Simulink.Signal` object is set, the `Alias` value is used as the name of the corresponding `CHARACTERISTIC` or `MEASUREMENT` A2L file entry. **Note** If you want to use the dSPACE-specific MATLAB configuration structure to configure a Simulink object for which the `Alias` property is set, you have to specify the `Alias` value as the `<ObjectName>` (instead of the original Simulink object name) in the `DSPACE_Config.<ModelName>.VARIABLES.<ObjectName>.PROPERTIES` structure element.
`Data Type`	If the `Data Type` property of a `Simulink.Parameter` or `Simulink.Signal` is set to `Boolean`, the `BIT_MASK` A2L file entry is set to the `0x01` default value.
`Description`	If the `Description` property of a `Simulink.Parameter` or `Simulink.Signal` is set, its value is used as the `LongIdentifier` value of the corresponding `CHARACTERISTIC` or `MEASUREMENT` A2L file entry. **Note** If the dSPACE-specific MATLAB configuration structure contains a value for the `LongIdentifier` of a `CHARACTERISTIC` or `MEASUREMENT` A2L file entry, this value is used instead of the `Description` property of the related Simulink objects.

Property of Simulink Objects	Effect on CHARACTERISTIC / MEASUREMENT A2L File Entries
`Min` and `Max`	If the `Min` and `Max` properties of a `Simulink.Parameter` or `Simulink.Signal` are set, their values are used as the `LowerLimit` and `UpperLimit` values of the corresponding `CHARACTERISTIC` or `MEASUREMENT` A2L file entry. **Note** If the dSPACE-specific MATLAB configuration structure contains values for the `LowerLimit` and `UpperLimit` of a `CHARACTERISTIC` or `MEASUREMENT` A2L file entry, these values are used instead of the `Min` and `Max` properties of the related Simulink objects.

Example of a parameter and a signal The following is an A2L file extract for a parameter and a signal of the `FLOAT64_IEEE` data type.

```
/begin CHARACTERISTIC
   /* Name */                  parameter
   /* Long identifier */       ""
   /* Characteristic-type */   VALUE
   /* Address */               0x0004AC20
   /* Record layout */         Scalar_FLOAT64_IEEE
   /* Maxdiff (Not used) */    0
   /* Conversion method */     COMPU_METHOD_1
   /* Lower limit */           -1.7976e308
   /* Upper limit */           1.7976e308

/end CHARACTERISTIC
```

```
/begin MEASUREMENT
   /* Name */                    signal
   /* Long identifier */         ""
   /* Data type */               FLOAT64_IEEE
   /* Conversion method */       COMPU_METHOD_1
   /* Resolution (Not used) */   0
   /* Accuracy (Not used) */     0
   /* Lower limit */             -1.7976e308
   /* Upper limit */             1.7976e308
/end MEASUREMENT
```

The default data type for Simulink models is `FLOAT64_IEEE`.

Look-up tables

In an A2L file, look-up tables are represented by `CHARACTERISTIC` and `AXIS` entries.

- For fixed and non-tunable breakpoints of a Simulink look-up table, `FIX_AXIS` A2L file entries are generated.

  The generation of references to computation methods (`COMPU_METHOD`) is suppressed for `FIX_AXIS` A2L file entries if these references are not required.
- For tunable and shared breakpoints of a Simulink look-up table, `COM_AXIS` A2L file entries are generated.
- For tunable but not shared breakpoints of a Simulink look-up table, `STD_AXIS` A2L file entries are generated.

Input signals with label If the input signal(s) (i.e., the driving signal(s) of the corresponding Lookup Table block) are labeled, one `MEASUREMENT` variable is generated for each input signal of the look-up table. The working point is then calculated using the input signal(s) and the corresponding function value of the look-up table. The names of the driving signals are used to set the `Input quantity` parameters of the corresponding `AXIS_PTS` definition. The working point can be visualized in a measurement and calibration system.

Input signals without label If the input signals have no labels, the `Input quantity` parameters are set to `NO_INPUT_QUANTITY` , and no corresponding `MEASUREMENT` variables are generated for the input signals.

Example of a look-up table with a labeled input signal Suppose you have a 1-dimensional look-up table with an input signal labeled `InputMeasurement`. The generated `MEASUREMENT` variable in the A2L file will be:

```
/begin MEASUREMENT
      /* Name */                    InputMeasurement
      /* Long identifier */         ""
      /* Data type */               FLOAT64_IEEE
      /* Conversion method */       COMPU_METHOD_1
      /* Resolution (Not used) */   0
      /* Accuracy (Not used) */     0
      /* Lower limit */             -1.7976e308
      /* Upper limit */             1.7976e308
      /* Address */                 ECU_ADDRESS  0x000AF2D0
/end MEASUREMENT
```

For the parameter containing the X-axis points, an `AXIS_PTS` definition will be generated:

```
/begin AXIS_PTS
      /* Name */                    lut1d_axis
      /* Long identifier */         ""
      /* Address */                 0x000A72C8
      /* Input quantity */          InputMeasurement
      /* Record Layout */           Axis_FLOAT64_IEEE
      /* Maxdiff (Not used) */      0
      /* Conversion method */       COMPU_METHOD_1
      /* Max axis points */         4
      /* Lower limit */             -1.7976e308
      /* Upper limit */             1.7976e308
/end AXIS_PTS
```

The `Input quantity` parameter name of the `AXIS_PTS` definition gets the input signal label. The working point can be calculated using the `InputMeasurement` input signal and the corresponding function value of the look-up table.

Grouping of A2L file entries

A2L file entries are grouped (by the `GROUP` keyword) according to the graphical grouping of the related variables in the Simulink model.

Tip

Grouping of A2L file entries is also possible via the dSPACE-specific MATLAB configuration structure. See Basics on Configuring A2L File Generation on page 18.
Grouping via the dSPACE-specific MATLAB configuration structure uses the `FUNCTION` keyword. As a consequence, graphical grouping (using the `GROUP` keyword) and grouping via the configuration structure (using the `FUNCTION` keyword) can be used for the same model.

Related topics

HowTos

How to Generate an A2L File....29

Limitations for A2L File Generation

Introduction

A2L file generation based on Simulink® Coder™ has some limitations.

Limitations related to the Simulink model

The following limitations for A2L file generation are related to the Simulink model:

- When you generate an A2L file for a model containing referenced models, no structure information based on the dSPACE-specific MATLAB configuration structure via the `FUNCTION` keyword is generated.
- If code generation is skipped for a referenced or top model due to incremental code generation, changes to the dSPACE-specific MATLAB configuration structure are not taken into account for A2L file generation.
- The A2L file hierarchy, which is based on the `GROUP` keyword, always corresponds to the group information generated by Simulink® Coder™. This cannot be disabled, for example, to achieve a flat A2L file hierarchy.
- Calibration protocol- and interface-specific entries are inserted to the A2L file for the top model only.
- The generated A2L file does not contain task information variables.

Limitations related to specific blocks, parameters, and other objects of the Simulink model

The following limitations for A2L file generation are related to specific blocks, parameters, and other objects of the Simulink model:

- Neither `CHARACTERISTIC`s nor referenced elements are created for structured tunable parameters.
- Neither `CHARACTERISTIC` nor `MEASUREMENT` nor referenced A2L file entries are created for Simulink buses.
- Neither `CHARACTERISTIC` nor `MEASUREMENT` A2L file entries are created for variables with more than three dimensions.
- No `CHARACTERISTIC` A2L file entries are created for variables that are defined as `Simulink.LookupTable` or `Simulink.Breakpoint` objects in the MATLAB workspace.
- No `CHARACTERISTIC` A2L file entries for model arguments as scalar elements or breakpoint/table data of look-up tables are created.
- You can use a common axis for several look-up tables of the *same* dimension. However, you cannot use a common axis for a 1-D- and a 2-D-look-up table at the same time.

A

C

D

G

L

M

S

U