1

Generating ASAP2 Files

ASAP2 is a data definition standard proposed by the Association for Standardization of Automation and Measuring Systems (ASAM). ASAP2 is a standard description you use for data measurement, calibration, and diagnostic systems.

This section includes the following topics:

- **Overview.** Topics you should be familiar with before working with ASAP2 file generation.
- **Targets Supporting ASAP2.** Real-Time Workshop® targets with built-in ASAP2 support.
- **Defining ASAP2 Information.** Signal and parameter information from a Simulink model needed to create an ASAP2 file.
- **Generating an ASAP2 File.** Procedure for creating an ASAP2 file from a Simulink model.
- **Customizing an ASAP2 File.** Target Language Compiler™ (TLC) files you can change to customize the ASAP2 file generated from a Simulink model.
- **Structure of the ASAP2 File.** Summary of the parts of the ASAP2 file and the Target Language Compiler functions used to write each part.

Overview

The Real-Time Workshop lets you export an ASAP2 file containing information about your model during the code generation process.

To make use of ASAP2 file generation, you should become familiar with the following topics:

- ASAM and the ASAP2 standard and terminology. See the ASAM Web site at `http://www.asam.de`.
- Simulink data objects. Data objects are used to supply information not contained in the model. See *Using Simulink*.
- Storage and representation of signals and parameters in generated code. See "Parameters: Storage, Interfacing and Tuning" and "Signals: Storage, Optimization, and Interfacing" in the *Real-Time Workshop User's Guide*.
- Signal and parameter objects and their use in code generation. See "Simulink Data Objects and Code Generation" in the *Real-Time Workshop User's Guide*.

Targets Supporting ASAP2

The Real-Time Workshop provides two target configurations you can use to generate ASAP2 files. You can select either of these target configurations from the System Target File Browser.

- The **ASAM-ASAP2 Data Definition Target** lets you generate only an ASAP2 file, without building an executable.
- The **Real-Time Windows Embedded Coder** lets you generate an ASAP2 file as part of the code generation and build process.

Procedures for generating ASAP2 files via these targets are given in "Generating an ASAP2 File" on page 1-6.

Alternatively, you can add ASAP2 support to your own target by defining the TLC variable `GenerateASAP2` in your system target file, as shown in the following code example.

```
%assign GenerateASAP2 = 1
%include "codegenentry.tlc"
```

Note You must define `GenerateASAP2` before including `codegenentry.tlc`.

Defining ASAP2 Information

The ASAP2 file generation process requires information about your model's parameters and signals. Some of this information is contained in the model itself. The rest must be supplied by using Simulink data objects with the necessary properties.

The Real-Time Workshop provides two example data classes to assist you in providing the necessary information. The classes are:

- `ASAP2.Parameter`, a subclass of `Simulink.Parameter`
- `ASAP2.Signal`, a subclass of `Simulink.Signal`

This document refers to these as the *ASAP2 classes*, and to objects instantiated from these classes as *ASAP2 objects*. The ASAP2 class creation files are located in the directory *`matlabroot`*`/toolbox/rtw/targets/asap2/asap2`. To create ASAP2 objects, make sure that this directory is on the MATLAB path.

As with the built-in `Simulink.Parameter` and `Simulink.Signal` classes, we recommend that you create your own packages and classes rather than using the ASAP2 classes directly. To do this, copy and rename the directory *matlabroot*`/toolbox/rtw/targets/asap2/asap2/@ASAP2`, and modify the class creation files it contains. You can extend the ASAP2 classes if additional properties are required. For general information about extending data object classes, see *Using Simulink*.

The following table contains the minimum set of data attributes required for ASAP2 file generation. Some data attributes are defined in the model; others are supplied in the properties of ASAP2 objects.For attributes that are defined in `ASAP2.Parameter` or `ASAP2.Signal` objects, the table gives the associated property name.

Data Required for ASAP2 File Generation

Data Attribute	Defined In	Property Name
Data type	Model	Not applicable
Scaling (if fixed point data type)	Model	Not applicable
Name (Symbol)	Data object	Inherited from name of handle to the data object to which parameter or signal name resolves
Long identifier (Description)	Data object	`LongID_ASAP2`
Minimum allowable value	Data object	`PhysicalMin_ASAP2`
Maximum allowable value	Data object	`PhysicalMax_ASAP2`
Units	Data object	`Units_ASAP2`
Memory Address (optional)	Data object (see note below)	`MemoryAddress_ASAP2` (optional)

Note on the Memory Address Attribute. The Memory Address attribute, if known before code generation, can be defined in the data object. Otherwise, a placeholder string is inserted. You can replace the placeholder with the actual address by post-processing the generated file. See the file `matlabroot/toolbox/rtw/targets/asap2/asap2/asap2post.m` for an example.

Generating an ASAP2 File

You can generate an ASAP2 file from your model in one of the following ways:

- Use the Real-Time Windows Embedded Coder to generate an ASAP2 file as part of the code generation and build process.
- Use the ASAM-ASAP2 Data Definition Target to generate only an ASAP2 file, without building an executable.
- Add ASAP2 support to your own target (see "Targets Supporting ASAP2" on page 1-3).

This section discusses how to generate an ASAP2 file via the targets that have built-in ASAP2 support.

Generating ASAP2 Files via the Real-Time Windows Embedded Coder

The procedure for generating a model's data definition in ASAP2 format via the Real-Time Windows Embedded Coder is as follows:

1. Create the desired model. Use appropriate parameter names and signal labels to refer to `CHARACTERISTICS` and `MEASUREMENTS` respectively.

2. Define the relevant `ASAP2.Parameter` and `ASAP2.Signal` objects in the MATLAB workspace.

3. Configure the data objects to generate unstructured global storage declarations in the generated code by assigning one of the following storage classes to the `RTWInfo.StorageClass` property:

   - `ExportedGlobal`
   - `ImportedExtern`
   - `ImportedExternPointer`

4. Configure the other data object properties such as `LongID_ASAP2`, `PhysicalMin_ASAP2`, etc.

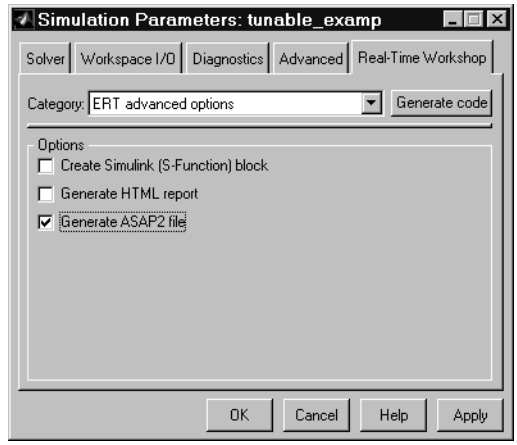
5. In the Advanced page of the **Simulation Parameters** dialog box, select the **Inline parameters** option.

   Note that you should *not* configure the parameters associated with your data objects in the **Model Parameter Configuration** dialog box. If a parameter that resolves to a Simulink data object is configured using the **Model**

Parameter Configuration dialog box, the dialog box configuration is ignored. You can, however, use the **Model Parameter Configuration** dialog to configure other parameters in your model.

6 In the Real-Time Workshop page, click **Browse** to open the System Target File Browser. In the browser, select the **Real-Time Windows Embedded Coder Target**.

7 Select **ERT advanced options** from the **Category** menu of the Real-Time Workshop page. Then select the **Generate ASAP2 file** option.



8 Click **Apply**.

9 Click **Build** (or G**enerate code**).

10 The Real-Time Workshop writes the ASAP2 file to the build directory. The ASAP2 filename is controlled by the ASAP2 setup file. By default, the file is named *model*`.a2l`.

Generating ASAP2 Files via the ASAM-ASAP2 Data Definition Target

The procedure for generating a model's data definition in ASAP2 format via the ASAM-ASAP2 Data Definition Target is as follows:

1. Create the desired model. Use appropriate parameter names and signal labels to refer to `CHARACTERISTICS` and `MEASUREMENTS` respectively.

2. Define the relevant `ASAP2.Parameter` and `ASAP2.Signal` objects in the MATLAB workspace.

3. Configure the data objects to generate unstructured global storage declarations in the generated code by assigning one of the following storage classes to the `RTWInfo.StorageClass` property:
   - `ExportedGlobal`
   - `ImportedExtern`
   - `ImportedExternPointer`

4. Configure the other data object properties such as `LongID_ASAP2`, `PhysicalMin_ASAP2`, etc.

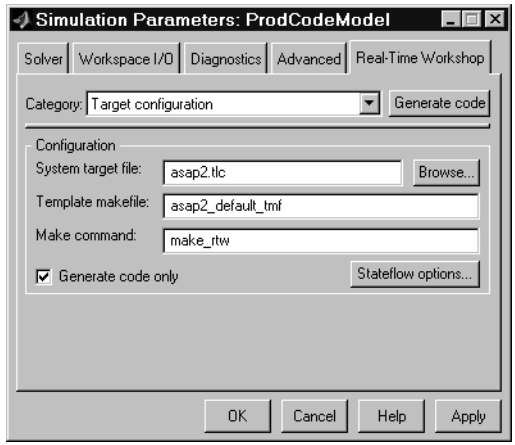
5. In the Advanced page of the **Simulation Parameters** dialog box, select the **Inline parameters** option.

   Note that you should *not* configure the parameters associated with your data objects in the **Model Parameter Configuration** dialog box. If a parameter that resolves to a Simulink data object is configured using the **Model Parameter Configuration** dialog box, the dialog box configuration is ignored. You can, however, use the **Model Parameter Configuration** dialog to configure other parameters in your model.

6. In the Real-Time Workshop page, click **Browse** to open the System Target File Browser. In the browser, select the **ASAM-ASAP2 Data Definition Target**.

7 Select **Target configuration** from the **Category** menu of the Real-Time Workshop page. Then select the **Generate code only** option.

This picture shows the correct configuration.



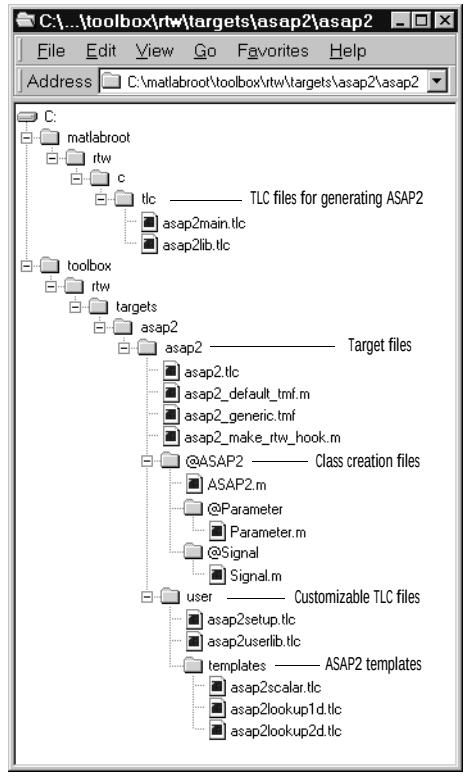
8 Click **Apply**.

9 Click **Generate code**.

10 The Real-Time Workshop writes the ASAP2 file to the build directory. The ASAP2 filename is controlled by the ASAP2 setup file. By default, the file is named *model*`.a2l`.

Customizing an ASAP2 File

The Real-Time Workshop Embedded Coder provides a number of TLC files to enable you to customize the ASAP2 file generated from a Simulink model. The following figure illustrates the hierarchy of ASAP2 related directories and files within the MATLAB directory.



ASAP2 File Structure on the MATLAB Path

The ASAP2 related files are located within the directories shown above. The files are organized as follows:

- TLC files for generating ASAP2 Files

  The *matlabroot*`/rtw/c/tlc` directory contains TLC files that generate ASAP2 files. These files are included by the Real-Time Workshop Embedded Coder and ASAP2 system target files (`ert.tlc` and `asap2.tlc`).

- ASAP2 target files

  The *matlabroot*`/toolbox/rtw/targets/asap2/asap2` directory contains the ASAP2 system target file and other control files.

- ASAP2 class creation files

  The *matlabroot*/toolbox/rtw/targets/asap2/asap2/@ASAP2 directory contains the M-files that define the ASAP2.Parameter and ASAP2.Signal classes.

- Customizable TLC files

  The *matlabroot*/toolbox/rtw/targets/asap2/asap2/user directory contains files that you can modify to customize the content of your ASAP2 files.

- ASAP2 templates

  The *matlabroot*/toolbox/rtw/targets/asap2/asap2/user/templates directory contains templates that define each type of CHARACTERISTIC in the ASAP2 file.

Customizing the Contents of the ASAP2 File

The ASAP2 related TLC files enable you to customize the appearance of the ASAP2 file generated from a Simulink model. Most customization is done by modifying or adding to the files contained in the *matlabroot*/toolbox/rtw/targets/asap2/asap2/user directory.This section refers to this directory as the asap2/user directory.

The user-customizable files provided are divided into two groups:

- The *static* files define the parts of the ASAP2 file that are related to the environment in which the generated code is used. They describe information specific to the user and/or project. The static files are not model-dependant.
- The *dynamic* files define the parts of the ASAP2 file that are generated based on the structure of the source model.

The procedure for customizing the ASAP2 file is as follows:

1. Make a copy of the asap2/user directory before making any modifications.

2. Remove the old asap2/user directory from the MATLAB path, or add the new asap2/user directory to the MATLAB path above the old directory. This will ensure that MATLAB uses the new ASAP2 setup file, asap2setup.tlc.

   asap2setup.tlc specifies which directories and files to include in the TLC path during the ASAP2 file generation process. Modify asap2setup.tlc to control the directories and folders included in the TLC path.

3 Modify the static parts of the ASAP2 file. These include:

- Project and header symbols, which are specified in `asap2setup.tlc`.
- Static sections of the file, such as file header and tail, `A2ML`, `MOD_COMMON`, etc. These are specified in `asap2userlib.tlc`.
- Specify the appearance of the dynamic contents of the ASAP2 file by modifying the existing ASAP2 templates, or by defining new ASAP2 templates. Sections of the ASAP2 file affected include:
- `RECORD_LAYOUTS`: modify appropriate parts of the ASAP2 template files.
- `CHARACTERISTICS`: modify appropriate parts of the ASAP2 template files.

  For more information on modifying the appearance of `CHARACTERISTICS`, see “ASAP2 Templates” on page 1-12.
- `MEASUREMENTS`: These are specified in `asap2userlib.tlc`
- `COMPU_METHODS`: These are specified in `asap2userlib.tlc`

ASAP2 Templates

The appearance of `CHARACTERISTICS` in the ASAP2 file is controlled using a different template for each type of `CHARACTERISTIC`. The `asap2/user` directory contains template definition files for scalars, 1-D Lookup Tables and 2-D Lookup Tables. You can modify these template definition files, or you can create additional templates as required.

The procedure for creating a new ASAP2 template is as follows:

1 Define a parameter group. See “Defining Parameter Groups”.

2 Create a template definition file. See “Creating Template Definition Files”.

3 Include the template definition file in the TLC path. The path is specified in the ASAP2 setup file, `asap2setup.tlc`.

Defining Parameter Groups. In some cases it is necessary for multiple parameters to be grouped together in the ASAP2 file (for example, the `x` and `y` data in a 1-D Lookup Table). Parameter groups enable Simulink blocks to define an associative relationship between some or all of their parameters. The following example shows the `Lookup1D` parameter group and describes how to create and use parameter groups in conjunction with the ASAP2 file generation process.

Parameter groups are created as part of the `BlockInstanceSetup` function within a block's TLC file. There are two built-in TLC functions that facilitate this process: `SLibCreateParameterGroup` and `SLibAddMember`. The following code fragment creates the `Lookup1D` parameter group in `look_up.tlc`. Similar syntax is used to create parameter groups for Look-Up Table (2D), FixPt Look-Up Table, and FixPt Look-Up Table (2-D).

```
%if GenerateInterfaceAPI
  %% Create a parameter group for ASAP2 data definition
  %assign group  = SLibCreateParameterGroup(block,"Lookup1D")
  %assign tmpVar = SLibAddMember(block,group,InputValues)
  %assign tmpVar = SLibAddMember(block,group,OutputValues)
%endif
```

`ParameterGroup` records are not written to the *model*`.rtw` file, but are included as part of the relevant `Block` records in the `CompiledModel`. The following code fragment shows the `Lookup1D` parameter group. The `Lookup1D` parameter group has two `Member` records. The `Reference` fields of these records refer to the relevant `x` and `y` data records in the `GlobalMemoryMap`.

```
Block {
  Type      Lookup
  Name      "<Root>/Look-Up Table"
  ...
  NumParameterGroups    1
  ParameterGroup {
    Name         Lookup1D
    NumMembers   2
    Member {
      NumMembers   0
      Reference    ...
    }
    Member {
      NumMembers   0
      Reference    ...
    }
  }
}
```

The `Lookup1D` parameter group is used by the function `ASAP2UserFcnWriteCharacteristic_Lookup1D`, which is defined in the

template definition file, asap2lookup1d.tlc. This function uses the parameter group to obtain the references to the associated x and y data records in the GlobalMemoryMap, as shown in the following code fragment.

```
%function ASAP2UserFcnWriteCharacteristic_Lookup1D(paramGroup) Output
  %assign xParam = paramGroup.Member[0].Reference
  %assign yParam = paramGroup.Member[1].Reference
  ...
%endfunction
```

Creating Template Definition Files. This section describes the components that make up an ASAP2 template definition file. This information is provided in the form code examples from asap2lookup1d.tlc, the template definition file for the Lookup1D template. This template corresponds to the Lookup1D parameter group.

Note When creating a new template, use the corresponding parameter group name in place of Lookup1D in the code fragments shown.

The following sections describe the components of an ASAP2 template definition file.

Template Registration Function. The input argument is the name of the parameter group associated with this template.

```
%<LibASAP2RegisterTemplate("Lookup1D")>
```

RECORD_LAYOUT Name Definition Function. Record layout names (aliases) can be arbitrarily specified for each data type. This function is used by the other components of this file.

```
%function ASAP2UserFcnRecordLayoutAlias_Lookup1D(dtId) void
  %switch dtId
  %case tSS_UINT8
    %return "Lookup1D_UBYTE"
  ...
  %endswitch
%endfunction
```

Function to Write RECORD_LAYOUT Definitions. This function writes out RECORD_LAYOUT definitions associated with this template. The function is

called by the built-in functions involved in the ASAP2 file generation process. The function name must be defined as shown, with the appropriate template name after the underscore.

```
%function ASAP2UserFcnWriteRecordLayout_Lookup1D() Output
  /begin RECORD_LAYOUT
%<ASAP2UserFcnRecordLayoutAlias_Lookup1D(tSS_UINT8)>
    ...
  /end   RECORD_LAYOUT
%endfunction
```

Function to Write the CHARACTERISTIC. This function writes out the `CHARACTERISTIC` associated with this template. The function is called by the built-in functions involved in the ASAP2 file generation process. The function name must be defined as shown, with the appropriate template name after the underscore.

The input argument to this function is a pointer to a parameter group record. The example shown is for a `Lookup1D` parameter group which has two members. The references to the associated `x` and `y` data records are obtained from the parameter group record as shown.

This function calls a number of built-in functions to obtain the required information. For example, `LibASAP2GetSymbol` returns the symbol (name) for the specified data record.

```
%function ASAP2UserFcnWriteCharacteristic_Lookup1D(paramGroup)
Output
  %assign xParam = paramGroup.Member[0].Reference
  %assign yParam = paramGroup.Member[1].Reference
  %assign dtId = LibASAP2GetDataTypeId(xParam)
    /begin CHARACTERISTIC
     /* Name */              %<LibASAP2GetSymbol(xParam)>
     /* Long identifier */   "%<LibASAP2GetLongID(xParam)>"
     ...
    /end CHARACTERISTIC
%endfunction
```

Structure of the ASAP2 File

The table below outlines the basic structure of the ASAP2 file and describes which TLC functions and files are used to create each part of the file.

- **Static parts** of the ASAP2 file are shown in **bold**.
- Function calls are indicated by `%<FunctionName()>`.

Sections of ASAP2 File and Related TLC Functions and Files

File Section	Contents of asap2main.tlc	TLC File Containing Function Definition
File header	%<ASAP2UserFcnWriteFileHead()>	asap2userlib.tlc
/**begin PROJECT** ""	**/begin PROJECT** "%<ASAP2ProjectName>"	asap2setup.tlc
/begin HEADER "" HEADER contents **/end HEADER**	**/begin HEADER** "%<ASAP2HeaderName>" %<ASAP2UserFcnWriteHeader()> **/end HEADER**	asap2setup.tlc asap2userlib.tlc
/begin MODULE "" MODULE contents: - A2ML - MOD_PAR - MOD_COMMON ...	**/begin MODULE** "%<ASAP2ModuleName>" %<ASAP2UserFcnWriteHardwareInterface()>	asap2setup.tlc asap2userlib.tlc
Model-dependent MODULE contents: - RECORD_LAYOUTs - CHARACTERISTICS - ParameterGroups - ModelParameters	%<SLibASAP2WriteDynamicContents()> Calls user-defined functions: ...WriteRecordLayout_*TemplateName*() ...WriteCharacteristic_*TemplateName*() ...WriteCharacteristic_Scalar()	asap2lib.tlc user/templates/...
- MEASUREMENTS - ExternalInputs - BlockOutputs	...WriteMeasurement()	asap2userlib.tlc
- COMPU_METHODS	...WriteCompuMethod()	asap2userlib.tlc
/end MODULE	**/end MODULE**	
/end PROJECT	**/end PROJECT**	
File footer/tail	%<ASAP2UserFcnWriteFileTail()>	asap2userlib.tlc