MATLAB Examples

Getting Started with Embedded Coder Support Package for AUTOSAR Standard

By MathWorks

Explore:
Embedded Coder

Try it in MATLAB

View in: Documentation

Generate AUTOSAR-compliant C code and export AUTOSAR XML (ARXML) descriptions from a Simulink® model.

Embedded Coder® software supports AUTomotive Open System ARchitecture (AUTOSAR), an open and standardized automotive software architecture. Automobile manufacturers, suppliers, and tool developers jointly develop AUTOSAR components. To develop AUTOSAR components in Simulink, follow this general workflow:

1. Create a Simulink representation of an AUTOSAR component.
2. Develop the component by refining the AUTOSAR configuration and creating algorithmic model content.
3. Generate ARXML descriptions and algorithmic C code for testing in Simulink or integration into the AUTOSAR Runtime Environment (RTE).

Contents

Prerequisites

The Embedded Coder Support Package for AUTOSAR Standard is required for working with the model in this example.

Install the AUTOSAR Standard Support Package

docid:matlab_external.budj4xt-2

Prepare Model for Code Generation

To see the steps for generating AUTOSAR-compliant C code and exporting ARXML descriptions from an AUTOSAR model, use the example model `rtwdemo_autosar_swc`.

Related Examples

Modeling Patterns for AUTOSAR Runnables

Use Simulink® models, sub
and functions to model AU
atomic software componen
their runnable entities (runr

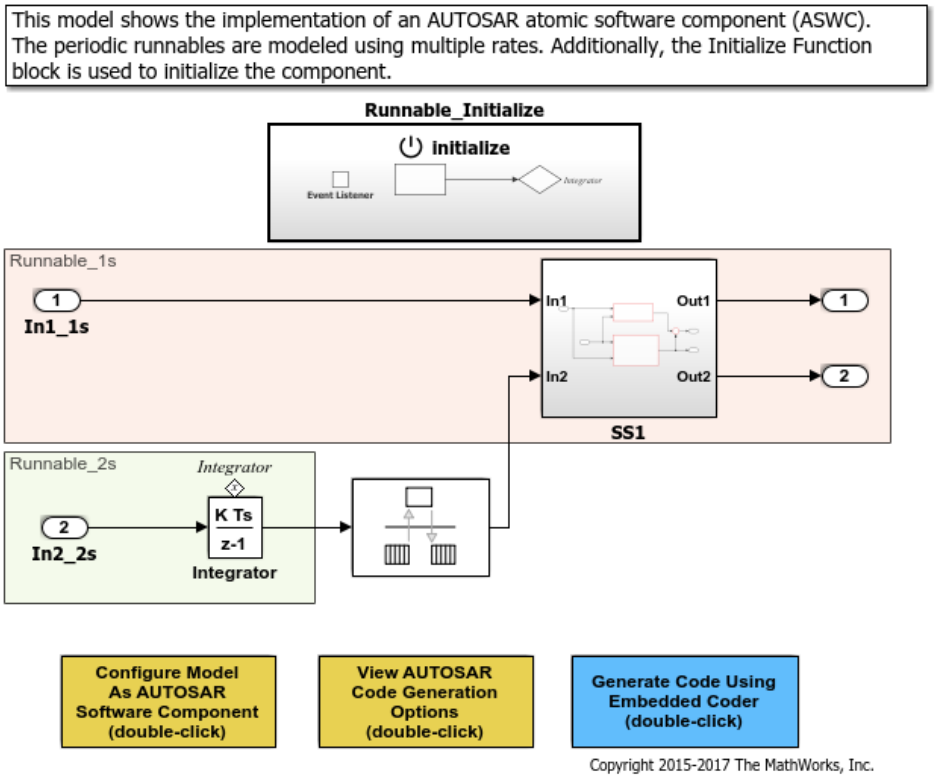
1. Open the model `rtwdemo_autosar_swc`.
2. Open the Configuration Parameters dialog box. Click the **Code** menu and select **C/C++ Code** > **Code Generation Options**.
3. Select AUTOSAR as the code generation target. In the **Code Generation** pane, change the **System target file** to `autosar.tlc`.
4. Specify an AUTOSAR schema version to use for ARXML file export. Go to the **Code Generation** > **AUTOSAR Code Generation Options** pane, and select a value for **Generate XML for schema version**. This example uses the value `4.0`.
5. Configure hardware settings for software-in-the-loop (SIL) testing and use them for both model builds in this example. Go to the **Code Generation** > **Verification** pane and select **Enable portable word sizes**. Go to the **Hardware Implementation** pane and select **Support long long**.
6. Click **Apply**.

An alternative method to prepare the model for code generation is to execute these commands.

```
% Model defines
modelName = 'rtwdemo_autosar_swc';

% Open the model
open_system(modelName);

% Programmatically set the system target file and AUTOSAR schema version
set_param(modelName,'SystemTargetFile','autosar.tlc');
set_param(modelName,'AutosarSchemaVersion','4.0');

% Hardware settings for software-in-the-loop simulation
set_param(modelName,'PortableWordSizes','on');
% Add long long support for production device Intel x86-64 (Windows64)
set_param(modelName,'ProdLongLongMode','on');
```

Getting Started with Embedded Coder Supp Package for AUTOSAR

Generate AUTOSAR-comp code and export AUTOSAF (ARXML) descriptions from Simulink® model. AUTOSA



Configure Simulink Representation of AUTOSAR Software Component

docid:ecoder_autosar.bt2wgsd

An AUTOSAR model provides a mapped configuration of an AUTOSAR software component. An AUTOSAR component is made up of elements defined in the AUTOSAR standard, such as ports, runnable entities, and events. The mapped configuration describes the AUTOSAR component and provides a mapping between Simulink model elements and AUTOSAR component elements.

To configure AUTOSAR properties and map Simulink model elements to AUTOSAR component elements, use the Configure AUTOSAR Interface dialog box.

1. Click the **Code** menu and select **C/C++ Code** > **Configure Model as AUTOSAR Component**.
2. Select **AUTOSAR Properties**. In the AUTOSAR Properties Explorer, you can configure AUTOSAR ports, runnables, events, inter-runnable variables, parameters, interfaces, computational methods, and XML options for the component.
3. Select **Simulink-AUTOSAR Mapping**. In the Simulink-AUTOSAR Mapping Explorer, you can configure the mapping of Simulink inports, outports, entry-point functions, data transfers, and lookup tables to

AUTOSAR elements.

4. To confirm that the AUTOSAR interface configuration meets validation requirements, click the **Validate** (check mark) button.

An alternative method to configure AUTOSAR properties and map Simulink model elements to AUTOSAR component elements is to use AUTOSAR property and map functions. For a listing of property and map functions, execute these commands.

```
help autosar.api.getAUTOSARProperties;
help autosar.api.getSimulinkMapping;
```

```
autosar.api.getAUTOSARProperties API class.

DATAOBJ = autosar.api.getAUTOSARProperties(MDLNAME) creates an object
DATAOBJ that can be used to query and set the AUTOSAR Properties of
Simulink model MDLNAME.

getAUTOSARProperties methods:
   add                      - add a property to an AUTOSAR element
   addPackageableElement    - add an AUTOSAR element to an AUTOSAR packa
   delete                   - delete an AUTOSAR element
   find                     - find AUTOSAR elements
   get                      - get the property of an AUTOSAR element
   set                      - set the property of an AUTOSAR element
   deleteUnmappedComponents - delete all unmapped components

See also autosar.api.create autosar.api.getSimulinkMapping

  Reference page in Doc Center
     doc autosar.api.getAUTOSARProperties


autosar.api.getSimulinkMapping API class.

MAPPINGOBJ = autosar.api.getSimulinkMapping(MDLNAME) creates an object
MAPPINGOBJ that can be used to query and set the Simulink mapping of
Simulink model MDLNAME
getSimulinkMapping methods:
   getDataTransfer   - get the mapping information for a Simulink data t
   getFunction       - get the mapping information for a Simulink entry
   getInport         - get the mapping information for a Simulink inport
   getOutport        - get the mapping information for a Simulink outpo
   getFunctionCaller - get the mapping information for a Simulink functi
   getLookupTable    - get the mapping information for a Simulink lookup
   getDataDefaults   - get the memory type configured for Signals or Dis
   mapDataTransfer   - map a Simulink data transfer line
   mapFunction       - map a Simulink entry point function
   mapInport         - map an inport
   mapOutport        - map an outport
   mapFunctionCaller - map a function caller
   mapLookupTable    - map a lookup table
   mapDataDefaults    - map a memory type for Signals or Discrete State

See also autosar.api.syncModel autosar.api.create autosar.api.getAUTOSAR

  Reference page in Doc Center
     doc autosar.api.getSimulinkMapping
```

Generate C Code and ARXML Descriptions

Building the model generates AUTOSAR-compliant C code and exports ARXML descriptions. Click the **Code** menu and select **C/C++ Code** > **Build Model**.

To see the results of the model build, examine the code generation report.

An alternative method to generate AUTOSAR-compliant C code and export ARXML descriptions from the model is to press **Ctrl+B**, or execute this command.

```
rtwbuild(modelName);
```

```
### Starting build procedure for model: rtwdemo_autosar_swc
### Generating XML files description for model: rtwdemo_autosar_swc
### Successful completion of code generation for model: rtwdemo_autosar_s
```

Verify AUTOSAR Code with Software-in-the-Loop Testing

The end goal of AUTOSAR model development and code generation is to integrate the generated C code and ARXML descriptions into automotive applications in the AUTOSAR Runtime Environment (RTE). An intermediate step to RTE integration is to verify the generated C code in Simulink by using software-in-the-loop (SIL) simulation. When you configure and run a SIL simulation for an AUTOSAR model, the AUTOSAR target automatically configures the generated code to route simulation data with AUTOSAR RTE API calls.

1. Configure the model for SIL simulation. Click the **Simulation** menu and select **Mode** > **Software-in-the-Loop (SIL)**.
2. To rebuild the model and run the SIL simulation, click the **Run** (right-arrow) button.
3. Open the Diagnostic Viewer to see log messages for the AUTOSAR model build and simulation.

An alternative method to configure and run the SIL simulation is to execute these commands.

```
% Configure the model for software-in-the-loop simulation
set_param(modelName,'SimulationMode','Software-in-the-loop');

% Run the simulation
silOut = sim(modelName,'ReturnWorkspaceOutputs','on');
```

```
### Starting build procedure for model: rtwdemo_autosar_swc
### Generating XML files description for model: rtwdemo_autosar_swc
### Successful completion of build procedure for model: rtwdemo_autosar_s
### Preparing to start SIL simulation ...
Building with 'Xcode with Clang'.
MEX completed successfully.
### Updating code generation report with SIL files ...
### Starting SIL simulation for component: rtwdemo_autosar_swc
### Stopping SIL simulation for component: rtwdemo_autosar_swc
```

Related Topics

docid:ecoder_doccenter.autosar-code-generation-maad

docid:ecoder_doccenter.autosar-software-components

mathworks.com