MATLAB EXPO 2017
KOREA
4월 27일, 서울
등록 하기 matlabexpo.co.kr
From Simulink to AUTOSAR Code

김종헌 부장
Senior Application Engineer
MathWorks Korea
Agenda

- Introduction to AUTOSAR Standards
  - Simulink approach to AUTOSAR

- AUTOSAR design workflows
  - Bottom Up
  - Top Down
  - Round trip
  - Overview:
    - Modeling AUTOSAR Components / Attributes
    - Components, Runnables and Events
    - Modeling styles
Complexity of Classic AUTOSAR is growing

AUTOSAR Releases

- Components specification
- Body and Interior Electronics APIs...
- APIs for Powertrain, Chassis...
- DCM, DEM Revisions...
- Added safety concepts...
- RTE Enhancements...
- Variant Handling for Application Interfaces...
- Efficient NV Data Handling via RTE...
- Crypto Interface, V2X support...

2.0  90 files
2.1  120 files
3.0  125 files
3.1  129 files
3.2  136 files
4.0  183 files
4.1  198 files
4.2  227 files
4.3  238 files

www.autosar.org
AUTOSAR Adoption
What is AUTOSAR?

AUTOSAR® (AUTomotive Open System ARchitecture) is an open and standardized automotive software architecture.
AUTOSAR Vision

AUTOSAR Slogan:

“Cooperate on Standards – compete on implementation”
AUTOSAR Members

9 Core Partners

- BMW Group
- Continental
- DAIMLER
- PSA PEUGEOT CITROËN
- OPEL
- TOYOTA
- VOLKSWAGEN AG

9 Development Members

- Itemis
- SGS
- C&S
- KERIAL
- OFFIS
- TÜV NORD
- VALIDAS

88 Associate Members
17 Attendees

57 Premium Members

- Freescale
- ARM

11 Development Members

- Autoliv
- HONDA
- HYUNDAI-KIA MOTORS
- KIA
- NISSAN

9 Core Partners

- BOSCH
- Ford
- MAN
- RENAULT
- VOLVO

Generic Tier 1

- Autoliv
- Denso
- Hitachi
- MBtech
- IBV GmbH
- KPI-T Cummins
- Altran Technologies
- Infineon
- NEC
- Toshiba
- Fujitsu

Standard Software

- Magna
- Valeo
- ZF Lenksysteme
- Vector
- TTTech
- IBM
- Renesas
- ST

Tools and Services

- MathWorks
- TietoEnator
- NXP
- Infineon
- NEC
- NEC

Semi-conductors

- ARM
- Freescale
- Toshiba
- Fujitsu

General OEM

- Autoliv
- Denso
- Hitachi
- MBtech
- IBV GmbH
- KPI-T Cummins
- Altran Technologies
- Infineon
- NEC
- NEC

www.autosar.org
AUTOSAR Support from Embedded Coder and Simulink

Software Architecture Definition

Application Layer

Run Time Environment (RTE)

Basic Software

Services Layer

ECU Abstraction Layer

Microcontroller Abstraction Layer

Complex Device Drivers

ECU Hardware

Modeling and Simulation

BSW Configuration & RTE Generation

Basic SW Providers

Authoring Tools

Behavior Modeling & Code Generation
Simulink Approach to AUTOSAR

Simulink and Embedded Coder + AUTOSAR Support package for Embedded Coder

- No separate AUTOSAR Blockset needed
- Available via web download
- Import
- Export
- Code-generation through Mapping
- Use the Simulink-to-AUTOSAR Mapping Explorer to configure the mapping of Simulink models to AUTOSAR software components.

C Code and ARXML

Export

Generate XML file for schema version

Maximum SHORT-NAME length: 32
- Use AUTOSAR compiler abstraction
- Support root-level matrix I/O using

Available via web download

3.0
3.1
3.2
4.0
4.1
4.2

No separate AUTOSAR Blockset needed
Supported AUTOSAR Design Workflows

Top-Down Workflow

Autosar

Authoring Tool

Export ARXML

Import/Update

Bottom-Up Workflow

Round-Trip Workflow

SIL/PIL Test

Simulink

C Code

Embedded Coder

Import/Update

Export ARXML

Autosar

SW-C Description

Autosar

SW-C Description

Autosar

SW-C Description
Modeling AUTOSAR Communication

- Ports in a AUTOSAR software component allow for communication

- Categories of ports based on direction
  - Require port
  - Provide port

- Each port can have either of the following Interfaces
Runnable Entities

- Each AUTOSAR SW-C is composed by one or more runnables/runnable entities
Supported Events for a Runnable
Each Runnable should have at least one event attached

- **TimingEvent** - Periodically scheduled Runnables
- **DataReceivedEvent** - Trigger the runnable when data is received
- **ModeSwitchEvent** - Triggered onEntry, onExit, or onTransition
- **OperationInvokedEvent** - Client-server type event
- **InitEvent** - designate an AUTOSAR runnable as an initialization runnable, and then map an initialization function to the runnable.
- **DataReceiveErrorEvent** - when the communication layer reports an error in data reception by the receiver component
- **ExternalTriggerOccuredEvent** – used to activate a runnable in an SWC as result of an explicit trigger by a runnable entity of some other SWC
Mapping Simulink to AUTOSAR

AR: Port

AR: Interface

AR: Data Element 1

AR: Data Element 2

AR: Data Element 3
Example Mapping to a Receiver Port

**Add a New Simulink Port**

![Diagram showing the addition of a new Simulink port](image)

- **Runnable3**
- **RPort_DE2**
- **RPort_DE3**

**Add a New Data Element to the Interface**

![Diagram showing the addition of a data element](image)

- **DE1**
- **DE2**
- **DE3**

**Map Simulink Port to AUTOSAR Port**

![Diagram showing the mapping process](image)

- **MRPort (ECU mode)**
- **RPort_DE1**
- **RPort_DE2**
- **RPort_DE3**

**Validate AUTOSAR to Simulink Mapping**

![Diagram showing the validation process](image)
Using MATLAB for automating common tasks

%% Setup AUTOSAR Configuration programmatically

model = 'Average_VehicleSpeed_Calculation';

% Modify AUTOSAR Properties
autosarProps = autosar.api.getAUTOSARProperties(model);
set(autosarProps, 'Input', 'IsService', true);
set(autosarProps, 'XmlOptions', 'ArxmlFilePackaging', 'SingleFile');

% Modify Simulink Mapping to AUTOSAR
slMap = autosar.api.getSimulinkMapping(model);
mapInport(slMap, 'Input', 'Input', 'Input', 'ExplicitReceive');
mapOutport(slMap, 'Output', 'Output', 'Output', 'ExplicitSend');
For the multi-runnable modeling pattern, each input port that represents a runnable trigger event will need to have the check box for Output function call checked as part of the inport properties.

This allows for function triggers from a test harness model to be passed across a model reference boundary.
Model AUTOSAR Components

**Application Layer**

- **AUTOSAR Software Component 1**
- ... (repeated)
- **AUTOSAR Software Component n**

**Virtual Functional Bus (VFB)**

**Periodic rate-based**

**Periodic and Asynchronous**

**Multi-rate and Asynchronous**
Bottom-Up Workflow (Starting from Simulink)

AUTOSAR Authoring Tool

Import SWC Description

Export SWC Description/Generate SWC C code

AUTOSAR design (Meta-model)
Launch AUTOSAR Configuration

- The method for Configuring AUTOSAR Properties is
  - selecting the Code Menu and then selecting C/C++ Code
  - Configure Model as an AUTOSAR Component. (This will bring up a dialog screen as shown.)

```
>> rtwdemo_autosar_counter
```

Configure AUTOSAR Interface (User Dialog)
The AUTOSAR Interface Configuration is split between two areas: Simulink-AUTOSAR Mapping and AUTOSAR Properties. See this accordion based UI control at the bottom left corner of the dialog screen.

With the accordion control set to Simulink-AUTOSAR Mapping, configure AUTOSAR Mapping of Simulink:
- Inports
- Outports
- Entry Point Functions
- Data Transfers (IRV – InterRunnable Variables)

The Green Check mark is to Validate the AUTOSAR Configuration.

The blue Circle arrow is to refresh the Simulink elements.
Editing AUTOSAR Properties

View / Edit AUTOSAR Properties and Simulink Mappings

With the accordion control set to AUTOSAR Properties, the user can configure AUTOSAR elements / attributes such as:

- Add / Remove / Edit AUTOSAR Entities such as Components / Ports and Interfaces
- Configure ARXML options such as modular or single file generated on export or build; package paths; allow or not allow implementation types.
Mapping Inports to AUTOSAR Receiver Ports

Here is an example of mapping a Simulink Inport to a AUTOSAR Port Data Element. In this case, you can see the Simulink Inport called Input is mapped to the AR:Port of Input. The actual selection of the AR:Element is a bit hidden in this view. Notice that the AR:DataAccessMode is also available for the user to select from this dialog.
Here is an example of mapping a Simulink Inport to a AUTOSAR Port Data Element. In this case, you can see the Simulink Inport called Input is mapped to the AR:Port of Input. The actual selection of the AR:Element is a bit hidden in this view. Notice that the AR:DataAccessMode is also available for the user to select from this dialog.
Here is an example of mapping a Simulink Function Entry Point to an AUTOSAR Runnable. Note: The Initialize Function most likely will get mapped to the Runnable_Step (In this example). This Initialize Function is for blocks within the Software Component that need initialization such as 1/z blocks. This is not the same as user initialization of signals or variables.
Generate Code

Build model into AUTOSAR compliant Code (Control B or build Icon show to the left).

- Generates both C Code & AUTOSAR Software Component description files (ARXML files).
- Code uses RTE APIs to access AUTOSAR Ports such as Sender Receiver ports or Client / Server Ports as needed.
Select AUTOSAR Target

- The mechanism for selecting an AUTOSAR target is similar to selecting an ERT target
  - via the Simulink Configuration Parameters menu, Code Generation Tab, System target file.
Overview of Generated ARXML

- Generated ARXML contains
  - Component and Internal Behavior
  - Datatypes
  - Implementation information
    - Lists all generated source code and ARXML files
  - Interfaces
  - Other entities
AUTOSAR Schema Versions

- Seamless support for AUTOSAR Releases
  - Import detects AUTOSAR 2.x – 4.x release from arxml file
  - User selects AUTOSAR release from configuration set options for code generation and arxml export

<table>
<thead>
<tr>
<th>MATLAB Release</th>
<th>AUTOSAR Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2015b, R2016a/b, R2017a</td>
<td>2.1, 3.0, 3.1, <strong>3.2</strong> (Rev 3.2.2), 4.0, 4.1, <strong>4.2</strong> (Rev 4.2.1, 4.2.2)</td>
</tr>
<tr>
<td>R2014b, R2015a</td>
<td>2.1, 3.0, 3.1, 3.2, 4.0, <strong>4.1</strong> (Rev 4.1.1)</td>
</tr>
<tr>
<td>R2012a/b, R2013a/b, R2014a</td>
<td>2.1, 3.0, 3.1, 3.2, <strong>4.0</strong> (Rev 4.0.2)</td>
</tr>
<tr>
<td>R2011b</td>
<td>2.0, 2.1, 3.0, 3.1, 3.2</td>
</tr>
<tr>
<td>R2010a/b, R2011a</td>
<td>2.0, 2.1, 3.0, 3.1</td>
</tr>
<tr>
<td>R2009a/b</td>
<td>2.0, 2.1, 3.0</td>
</tr>
<tr>
<td>R2008a/b</td>
<td>2.0, 2.1</td>
</tr>
</tbody>
</table>
MISRA C:2012 for AUTOSAR target

- 100% Compliance with MISRA C:2012 Mandatory and Required rules
Verification with Software- and Processor-In-The-Loop (PIL)

- Support for SIL/PIL with AUTOSAR target
- Profile code and measure execution time on target
- Develop a custom PIL target for AUTOSAR using the toolchain build approach
Top-Down Workflow (Starting from SWC Description)

AUTOSAR Authoring Tool

Import SWC Description

Export SWC Description

Merge SWC Description

Model Based Design

Export SWC Description/Generate SWC C code
Top Down Workflow

**AUTOSAR Authoring Tool**

**ARXML Files**

- **Import as new Simulink model**
- **Update existing Simulink model**

Top Down Workflow

Starts with Authoring Tool, then user exports ARXML files from Authoring tool.

User can then either import the ARXML files into a new Simulink Skeleton model or Update an existing Simulink Model.
% Import ARXML Files
importerObj = arxml.importer('rtwdemo_autosar_multirunnables.arxml')

% Create new model with interfaces
model = importerObj.createComponentAsModel('/pkg/swc/ASWC');

Top Down Workflow Commands to create a skeleton model with no internal behavior setup in the skeleton model (no runnables)

Note: After running the arxml.importer command, here is the package that is needed for the second command of create Component as Model
Import with Internal Behavior

Top Down Workflow

Commands to create a skeleton model with internal behavior setup in the skeleton model (create with runnables)
Updating Existing Models from ARXML

V1.arxml  Updated to V2.arxml
Update Existing Models from ARXML

Top Down Workflow
Commands to update an existing Simulink model that already has an AUTOSAR configuration.

Notice that a report was created such that the user can understand what has changed in the model.
% cleanup
bdclose('all');
clear;

% Import ARXML Files
Obj = arxml.importer('SBRLogic.swc.arxml');

% Create new model with interfaces and internal behavior
model = Obj.createComponentAsModel('/SBRLogic_pkg/SBRLogic_swc.1xml');
AUTOSAR Support from Embedded Coder

Author and develop AUTOSAR software components for automotive systems

AUTOSAR (AUTomotive Open System ARchitecture) is an open and standardized automotive software architecture jointly developed by automobile manufacturers, suppliers, and tool developers.

Embedded Coder® Support Package for AUTOSAR Standard lets engineers model and simulate AUTOSAR software components, generate AUTOSAR production code, and verify AUTOSAR generated code using software- and processor-in-the-loop simulations. The support package also enables import and export of AUTOSAR Software Component descriptions that support top-down, bottom-up, and round-trip workflows involving third-party AUTOSAR authoring tools such as DaVinci Developer.

Platform and Release Support

See the hardware support package system requirements table for current and prior version, release, and platform availability.

View new features in the release notes.

http://www.mathworks.com/hardware-support/autosar.html
Thank You!