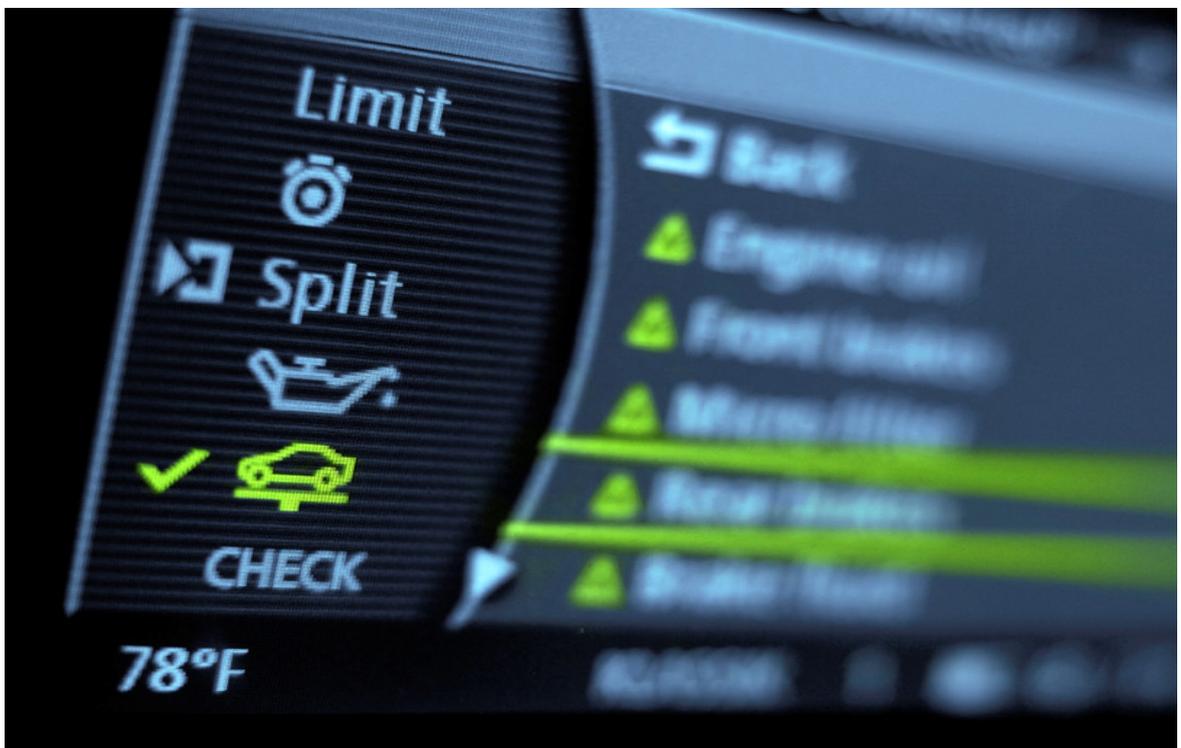




Elektrobit

SystemDesk - EB tresos Studio - TargetLink Workflow descriptions

Document date: 2018-01-22



dSPACE SystemDesk 4.8

EB tresos Studio for ACG8 (Version 23.0.0)

TargetLink 4.2 - AUTOSAR 4.3.0

Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
Phone: +49 9131 7701 0
Fax: +49 9131 7701 6333
E-mail: info.automotive@elektrobit.com

Technical support

Europe

Phone: +49 9131 7701 6060

Japan

Phone: +81 3 6421 7140

USA

Phone: +1 888 346 3813

China

Phone: +86 10 6781 7020-206

Support URL

<https://www.elektrobit.com/support>

Legal Notice

Confidential and proprietary information

ALL RIGHTS RESERVED. No part of this publication may be copied in any form, by photocopy, microfilm, retrieval system, or by any other means now known or hereafter invented without the prior written permission of Elektrobit Automotive GmbH.

ProOSEK®, tresos®, and street director® are registered trademarks of Elektrobit Automotive GmbH.

All brand names, trademarks and registered trademarks are property of their rightful owners and are used only for description.

© 2018 Elektrobit Automotive GmbH

Table of contents

1	Overview	5
2	SystemDesk and EB tresos Studio	7
2.1	Overview	7
2.2	Developing the architecture.....	8
2.2.1	Step 1: Importing the system or communication description	9
2.2.2	Step 2: Modeling the Software architecture	9
2.2.3	Step 3 (Optional): Modeling the system	16
2.2.4	Step 4: Exporting a system extract.....	17
2.3	Configuration and generation of basic software.....	18
2.3.1	Overview	19
2.3.2	Creating a new ECU configuration project	21
2.3.3	Importing the system extract	26
2.3.4	Configuration of AUTOSAR modules	31
2.3.5	Verifying the configuration.....	42
2.3.6	Adding the communication parts to the basic software	43
2.3.7	Export of the service component interfaces	43
2.3.8	Quick test building of generated software	44
3	Function development.....	45
3.1	Overview	45
3.2	Exporting a software component from SystemDesk	46
3.3	Importing a software component in TargetLink	48
3.4	AUTOSAR support in TargetLink.....	49
3.4.1	Frame model generation	49
3.4.2	Generation of application data types or implementation data types	50
3.4.3	Generating a constant specification mapping	51
3.5	Let TargetLink automatically create the implementation constants and the mappings.Delivering of the software component and the code files.....	52
4	Building the ECU target software.....	53
4.1	Preparing the build environment.....	53
4.2	Building the software.....	54
5	Runnable mappings exchange	55
5.1	Creating runnable mappings in SystemDesk	56
5.1.1	Step 1: Importing EB tresos Studio pa-rameter definitions into SystemDesk.....	56
5.1.2	Step 2: Creating RTE and OS module configurations	57
5.1.3	Step 3: Unifying the ECU extract top level composition	59
5.1.4	Step 4: Exporting the OS and RTE nodule configurations	63
5.2	Merging the runnable mappings into EB tresos Studio	64
5.2.1	Initial folder setup and batch file adjustment	65
5.2.2	Initial creation of the EB tresos Studio importers	67
5.2.3	Executing the MergeMappings.bat and Run Importer	69

6	Detailed element descriptions in SystemDesk	71
6.1	Mandatory attributes.....	71
6.2	Data type mapping	73
6.3	Constant specification mapping.....	76

1 Overview

SystemDesk, TargetLink and EB tresos Studio are three AUTOSAR tools which you can use together in an AUTOSAR tool chain to create ECU software:

- SystemDesk to model the architecture of the application software
- TargetLink to generate code for the application software
- EB tresos Studio to configure and generate the basic software as well as for RTE configuration and generation.

To create executable code for the ECU, these three tools must work well together. This document explains in detail how to interconnect the tools and which steps of the workflow are performed in which tool.

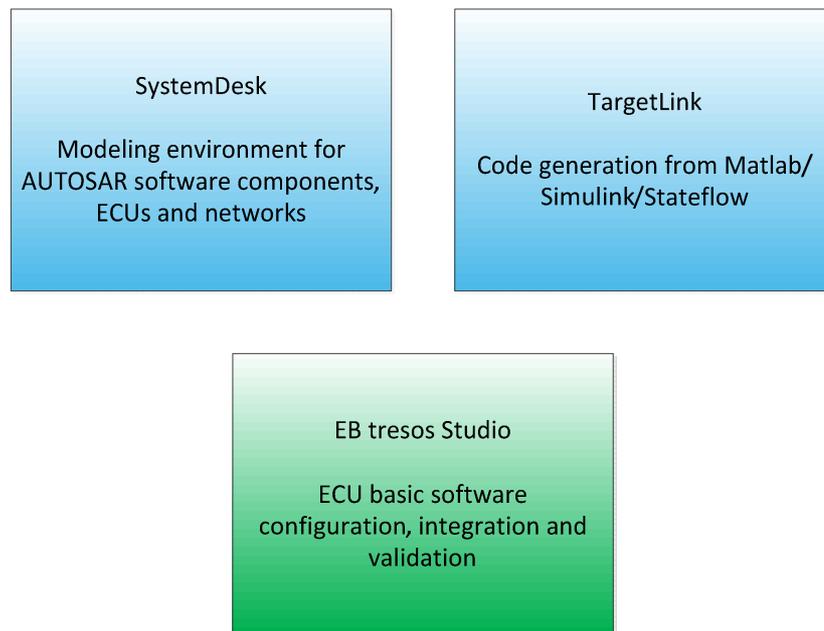


Figure 1: AUTOSAR tools described in this document

This document describes the interaction between the three AUTOSAR tools dSPACE SystemDesk, dSPACE TargetLink and, EB tresos Studio:

- SystemDesk 4.8
- EB tresos Studio 23.0.0
- TargetLink 4.2

It is assumed that you are familiar with using these tools, so no further introduction to them is provided here. You can find further information on the single tools in the following documentations:

- SystemDesk 4.x Tutorial
- SystemDesk 4.x Guide
- SystemDesk 4.x Reference
- ContainerManagementDocument
- TargetLink AUTOSAR Modeling Guide
- EB tresos Studio user's guide

This application note first describes the interfaces of the three tools and how to work with them in order to get the ECU target software.

- Chapter 2 is dedicated to the interface between SystemDesk and EB tresos Studio.
- Chapter 3 explains how to integrate TargetLink into the tool chain.
- Chapter 4 describes how to build the ECU target software.

The subsequent chapters give help for the following topics:

- Chapter 5 describes how to configure OS tasks and runnable mappings in SystemDesk and exchange with EB tresos Studio.
- Chapter 6 describes how to model specific elements in SystemDesk to enable a smooth transfer of the data to EB tresos Studio.

2 SystemDesk and EB tresos Studio

This chapter describes the components and features of EB tresos AutoCore Generic 7 Memory Stack (ACG7 Memory Stack) in detail.

2.1 Overview

This chapter describes the typical steps which are needed when you use SystemDesk and EB tresos Studio to create the software of an ECU. Figure 2 gives an overview of the complete workflow.

Step 1:

Import the system or communication description in SystemDesk which is provided by the OEM.

Step 2:

Use SystemDesk to model the software architecture and the system. For this task, it can also be necessary to import service interfaces, which need to be exported from EB tresos Studio.

Step 3:

When the architecture and system are complete, they are exported from SystemDesk and imported in EB tresos Studio. In EB tresos Studio you configure and generate the basic software which results in code files for the basic software.

This chapter describes the single steps of this workflow. Section 2.2 is dedicated to the steps you perform in SystemDesk, section 2.3 describes the steps in EB tresos Studio.

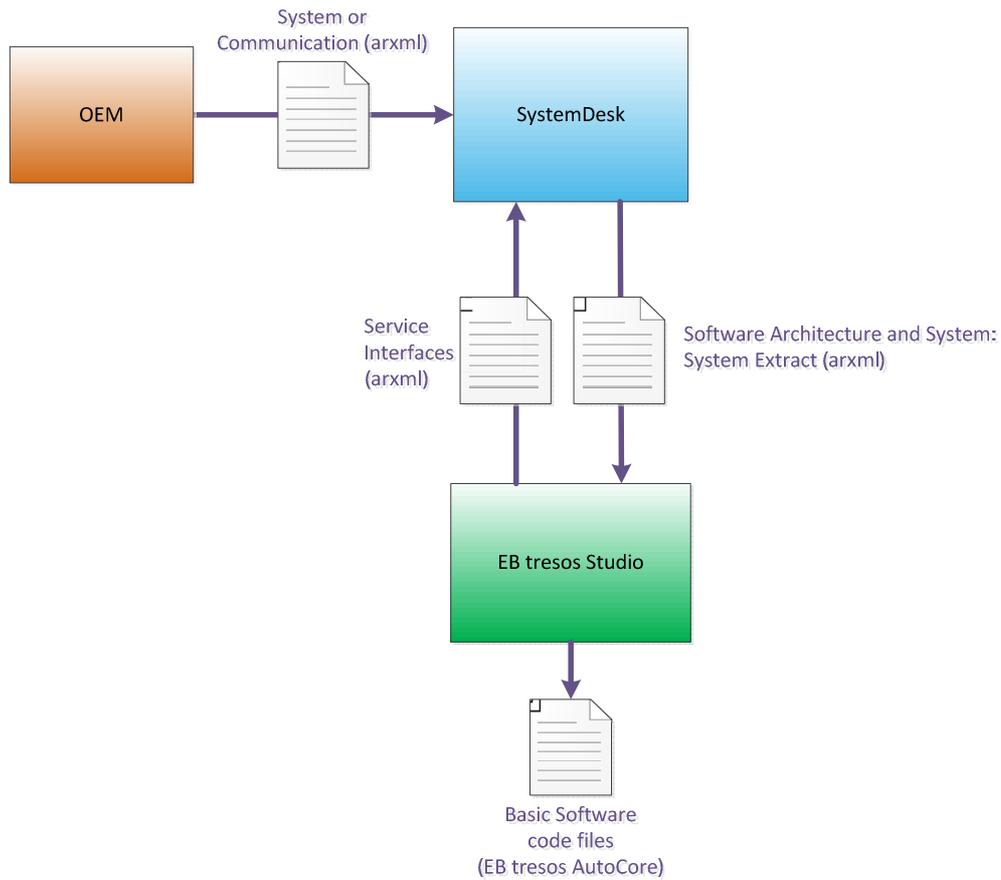


Figure 2: Overview of the workflow with SystemDesk and EB tresos Studio

The following sections describe the single steps of this workflow in detail. They also give hints about how to avoid common pitfalls.

2.2 Developing the architecture

This section describes those parts of the workflow which are performed in SystemDesk. It is divided into several steps from import of the system or communication description to export of the complete system. A detailed instruction on how to model the more complex elements is given in section 6.

2.2.1 Step 1: Importing the system or communication description

To import the system or communication description in SystemDesk, use the AUTOSAR import. Select **File / Import AUTOSAR**.

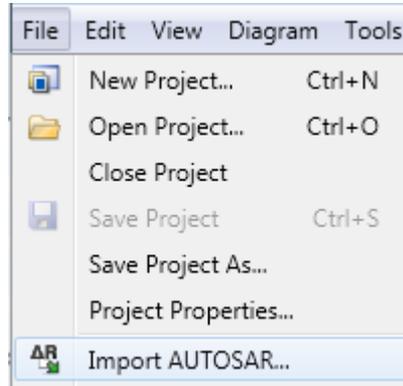


Figure 3: Importing a system or communication description

Select the ARXML file, click Open and then select the complete content of the ARXML file. When you click Finish Import, the complete content of the file is imported into SystemDesk. You can see it in the Project Manager.

2.2.2 Step 2: Modeling the Software architecture

The software architecture is modeled in SystemDesk. Using the AUTOSAR package structure, you can create software components with internal behaviors and their interconnections. For details about modeling the software architecture, see the SystemDesk Guide.

If the imported file already contains a composition for your software architecture, model your software architecture inside this composition. Otherwise, you need to create one composition which contains the complete architecture for your ECU with all software components and their interconnections. In the following, this composition is called the top-level composition.

When you model your architecture, make sure that it is correct and complete. The following sections describe important aspects for modeling the software architecture.

The modeling guideline takes the later generation phase of the RTE and the basic software into account. SystemDesk provides model validation with a variety of predefined rule sets for a specific task. It is recommended to set the validation rule set to "EB tresos compatibility check". Details on this topic are described in section 2.2.2.5.

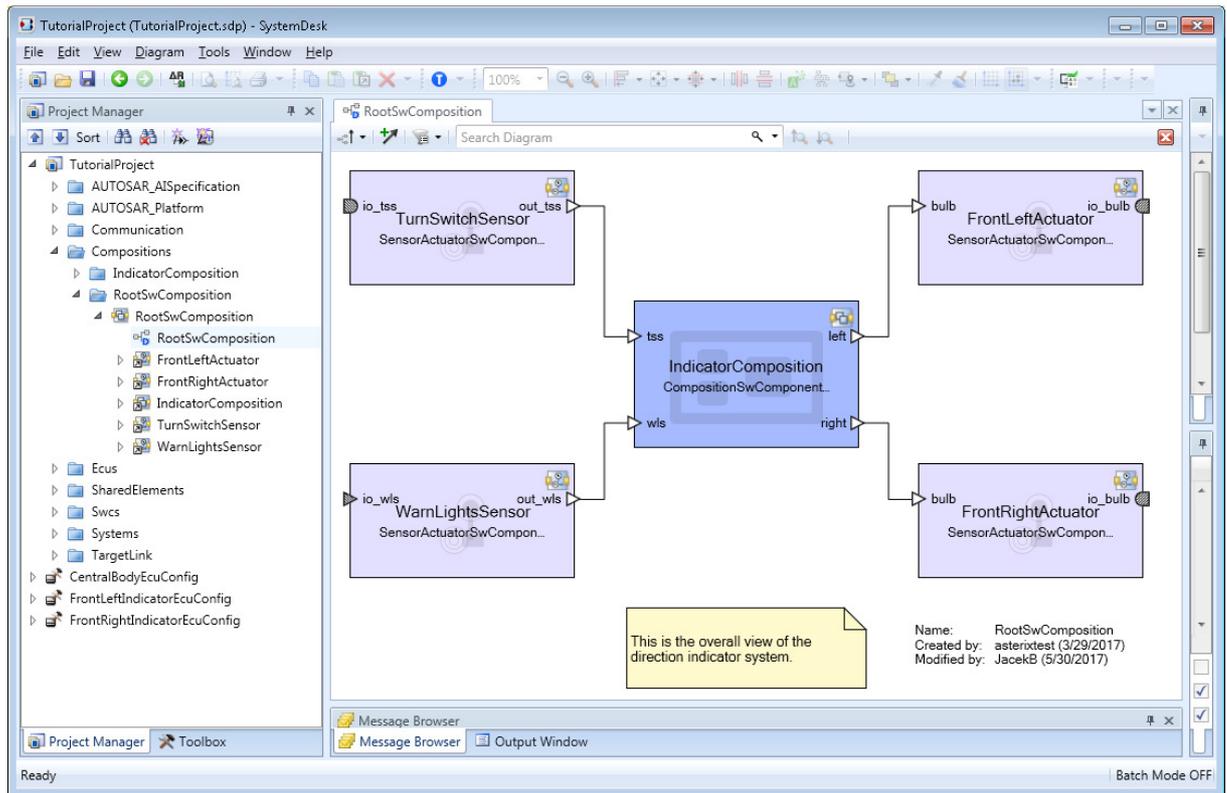


Figure 4: Project with a complete top-level composition

2.2.2.1 Service interfaces

In AUTOSAR, there are some basic software modules that have an interface to the application components (for example Dem and EcuM). Such modules are also called service components, because they make specific services available to the application components. The actual specification of the service components is always project-dependent. First, you must configure a module of this kind in EB tresos Studio, then you can generate the service component description, and finally, you can connect the ports of the service components to the ports of the application components. You can perform all these steps in EB tresos Studio. For more information, see section 2.3.6.

You must ensure that the ports of the application components are compatible with the ports of the service components, as otherwise RTE generation is not possible. This means that the interface descriptions of the services are required during the creation of the software architecture in SystemDesk.

EB tresos Studio therefore lets you export all the interface descriptions of the service components in the basic software. For further details on the service components and their export in EB tresos Studio, see section 2.3.7.

You can now import these interface descriptions in SystemDesk, and use them for the ports of the application components.

2.2.2.2 Initial values

To generate the RTE, it is advisable that you define an initial value for each variable data prototype and parameter data prototype. Examples for variable data prototypes are interruptible variables or data elements in sender-receiver interfaces. If the data prototype is accessed, but it does not have a value yet, the initial value is used instead. For most kinds of variable data prototypes, you can define the initial values in their **Properties** dialog.

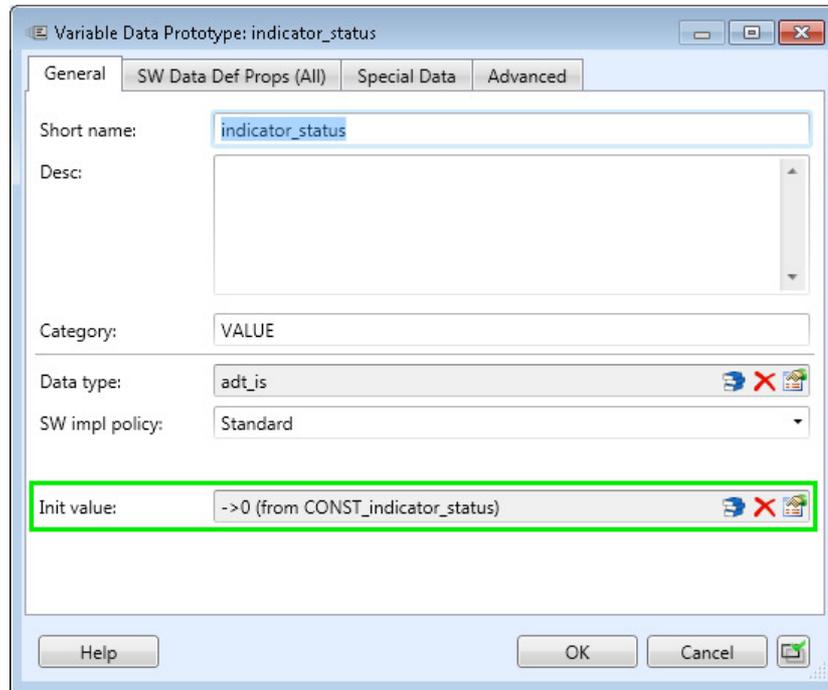


Figure 5: Defining initial values in the variable data prototype's Properties dialog

Exceptions are variable/parameter data prototypes defined in interfaces. Initial values for these are defined separately for each port in its com specs. For each connection between two ports which transmit a variable/parameter data prototype, the initial value(s) must be defined in either the provide-port or the require-port. If an initial value is defined in both ports, the initial value of the require-port is used.

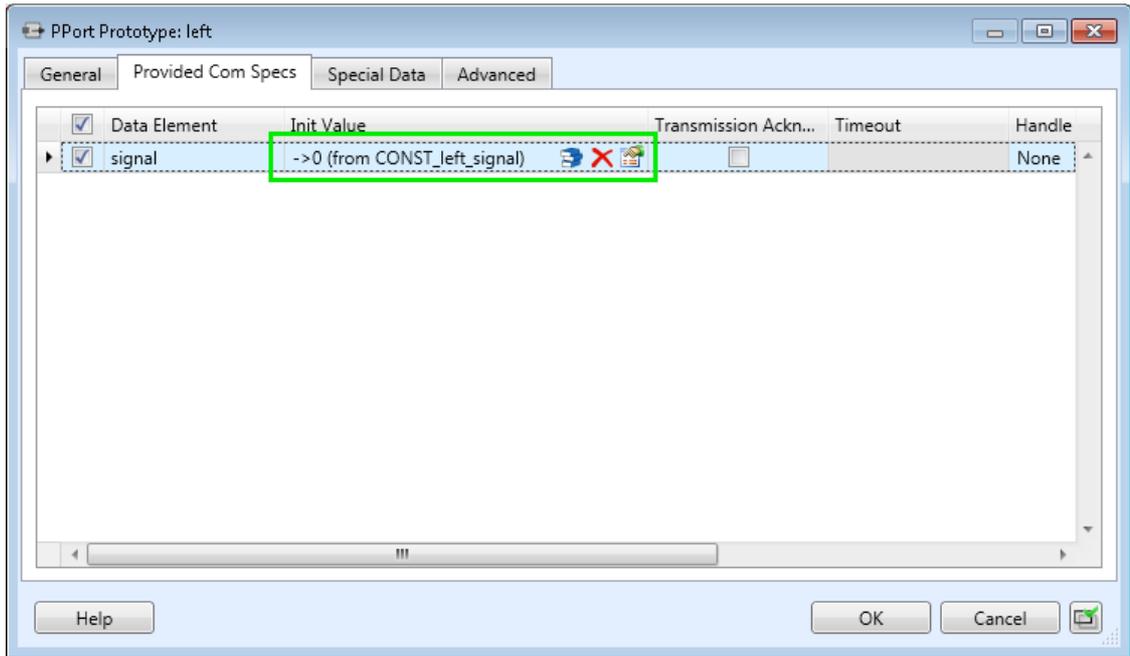


Figure 6: Defining initial values in the com spec of a port

2.2.2.3 Application data types and application constants

With release 4, AUTOSAR distinguishes application data types (ADTs) and implementation data types (IDTs). ADTs describe data as physical values from an application view, whereas IDTs represent the same data as internal values and specify the implementation details. IDTs are needed for code generation in TargetLink and also for RTE generation. ADTs must be mapped to IDTs. For this, you need one or more data type mapping sets, which you can create in SystemDesk via the context menu of a package. You can map the data types in its **Properties** dialog.

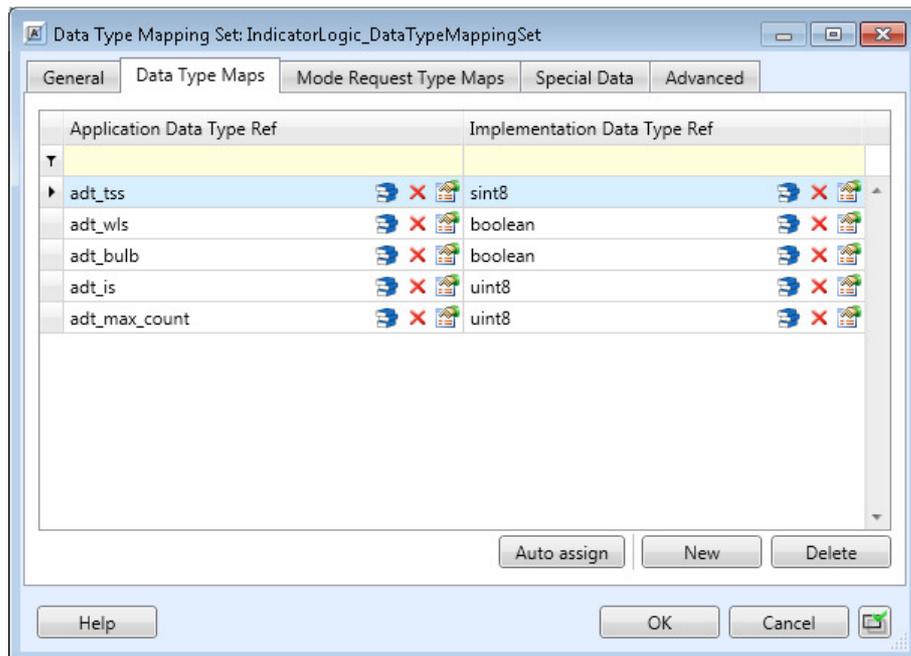


Figure 7: Mapping of application to implementation data types

You must create at least one reference to a data type mapping set from each internal behavior of an atomic software component. All ADTs which are used in a software component need to be mapped to IDTs in one of the data type mapping sets which are referenced by it. You can use the same mapping for several software components or you can create a different mapping for each SWC.

SystemDesk supports the automatic creation and mapping of IDTs to ADTs.

If you use TargetLink, you can also use the Implementation Data Type Creation Wizard to create the mappings. Details about how to model the data type mapping in SystemDesk are described in section 6.2. The usage of TargetLink's ImplementationDataType Creation Wizard is explained in section 3.4.2.

If you use application data types, you might additionally need to create one or more constant specification mappings.

This is the case if you have application constants in your project. Application constants describe a physical value, but for RTE generation the corresponding internal values are required. For this, EB tresos Studio expects a mapping of application constants to implementation constants which represent the corresponding internal values.

SystemDesk automatically creates application constants for values of application data types: For example, initial values for data elements typed by an application data type are application constants. You can perform the mapping in SystemDesk, see section 6.3 for a detailed instruction, or in TargetLink, see section 3.4.3.

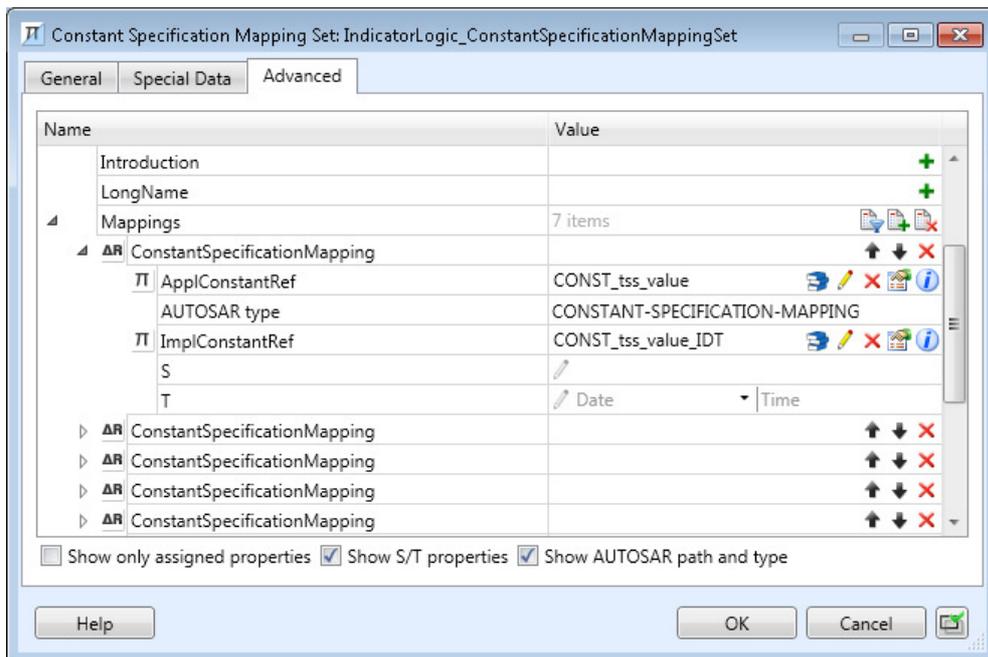


Figure 8: Constant specification mapping

2.2.2.4 SWC implementation

The element SWC implementation describes the implementation details of one internal behavior. For each internal behavior, you need at least one SWC implementation to be able to generate the RTE in EB tresos Studio. You can create it from the context menu of an internal behavior.

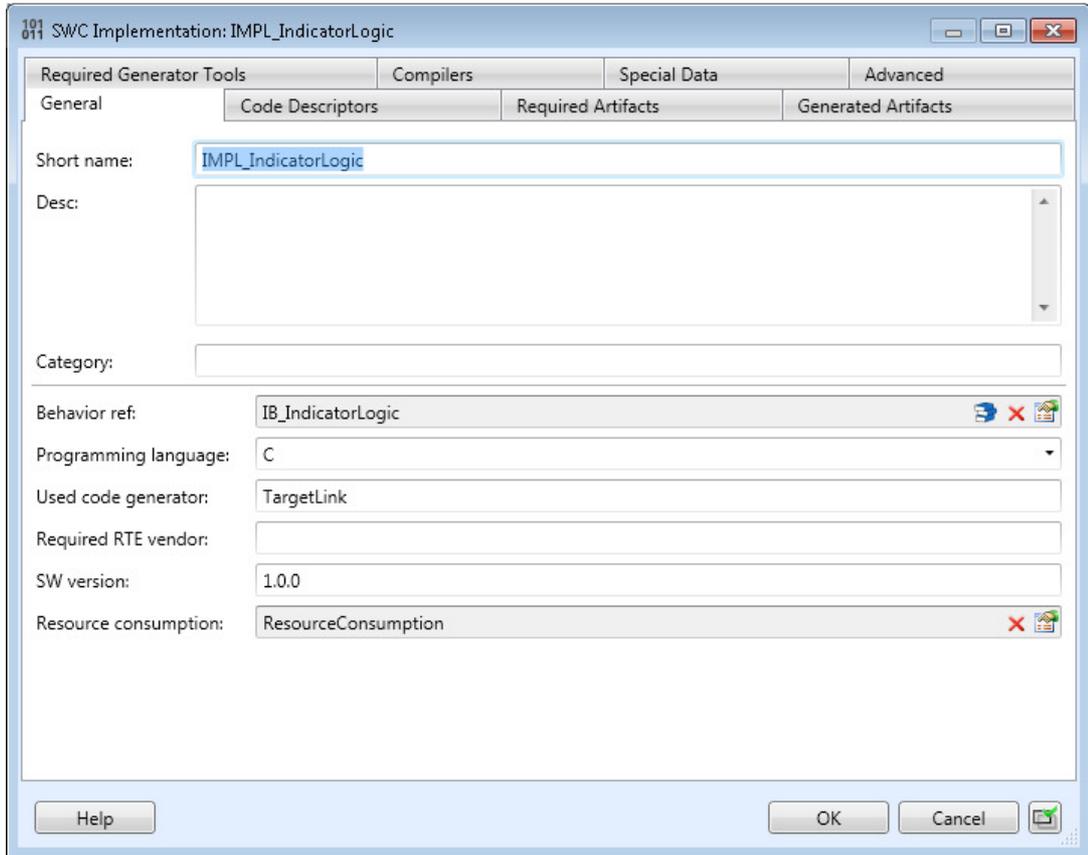


Figure 9: SWC implementation Properties dialog

EB tresos Studio requires at least one code descriptor with an artefact in it in order to accept the implementation element as valid. You can create the code descriptor in the **Properties** dialog of the implementation. There you can add the application code files to the code descriptor.

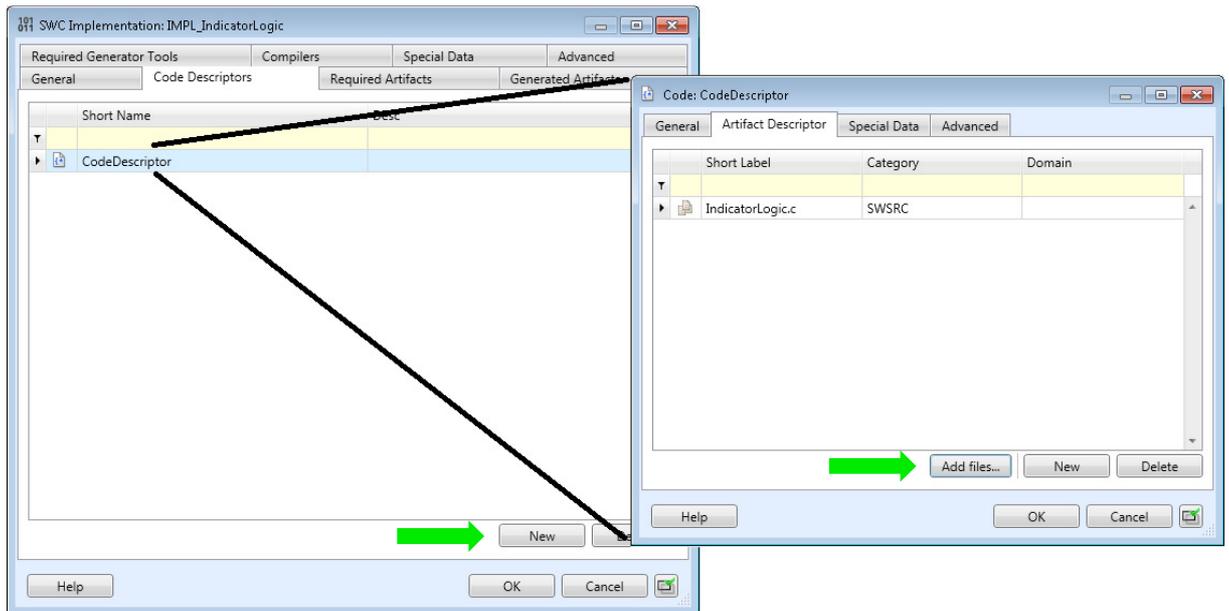


Figure 10: Describing code files

If you use TargetLink for code generation, you do not need to model the SWC implementations by hand: TargetLink creates them automatically. You only need to reimport the ARXML-description of the software component in SystemDesk after code generation, see section 3.5.

2.2.2.5 Validating

While modeling the software architecture, use SystemDesk's validation mechanism. The validation assists you in identifying some possible errors at this early stage.

You can use the default validation for checking general consistency constraints and rules defined by AUTOSAR, or you can use the predefined validation rule set EB tresos Compatibility Check.

When activated, SystemDesk additionally checks special issues regarding the exchange with EB tresos Studio. For example, it is important to see already in SystemDesk whether your project is complete or which model parts are still missing. Like this, you can detect already in SystemDesk whether your project can be processed in EB tresos Studio. However, validation in SystemDesk can identify most but not all potential problems that might be reported by EB tresos Studio in a later step.

You can activate the EB tresos Compatibility Check in SystemDesk's tool bar. After that, you can validate elements from their context menu in the Project Manager or directly from their Properties dialog.

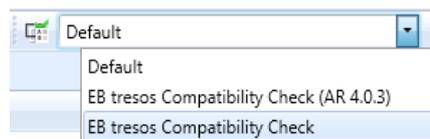


Figure 11: Activating the EB tresos Compatibility Check validation rule set

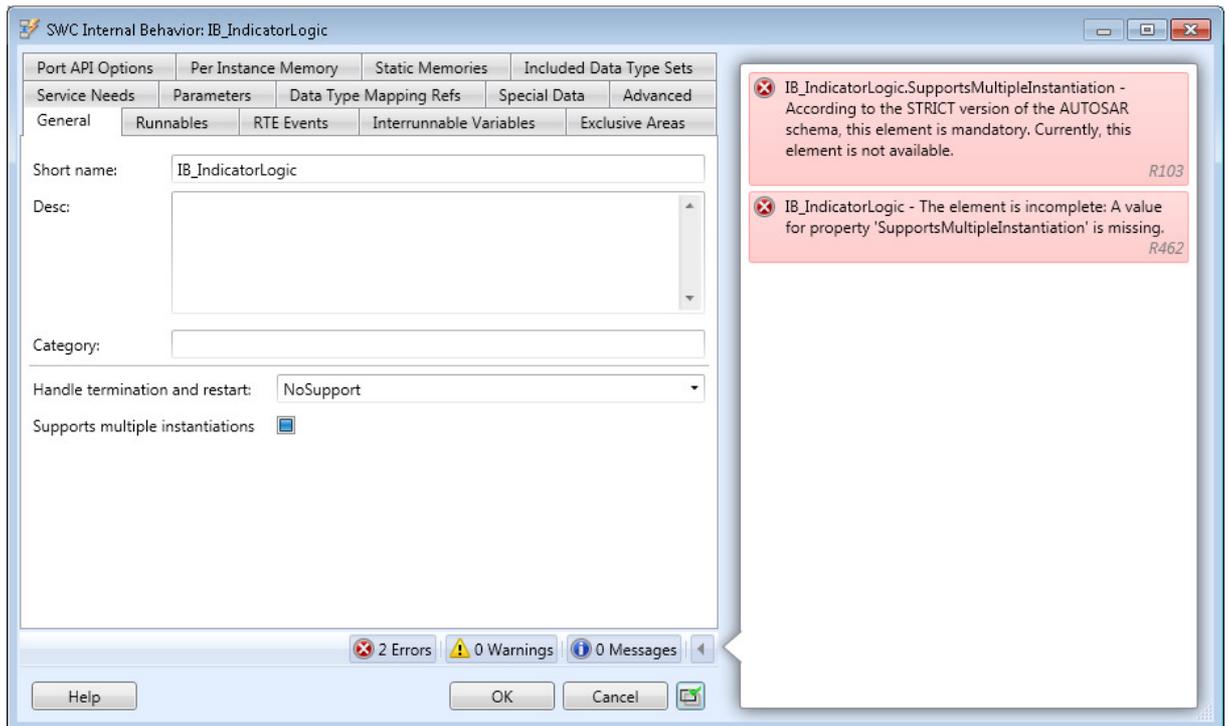


Figure 12: Validation of an internal behavior without a runnable

2.2.3 Step 3 (Optional): Modeling the system

Whether you need to model a system depends on what you have imported in the first step. If you have imported a complete system including a mapping of data elements to system signals, you do not need to do anything.

Otherwise, it is recommended to model the system in SystemDesk.

SystemDesk offers the System Manager for this task. The most important parts the system consists of are the root software composition, the ECU(s) and the imported network topology, a mapping of software components to ECUs and a mapping of data elements to system signals. For an introduction on how to model the system, see the SystemDesk Guide.

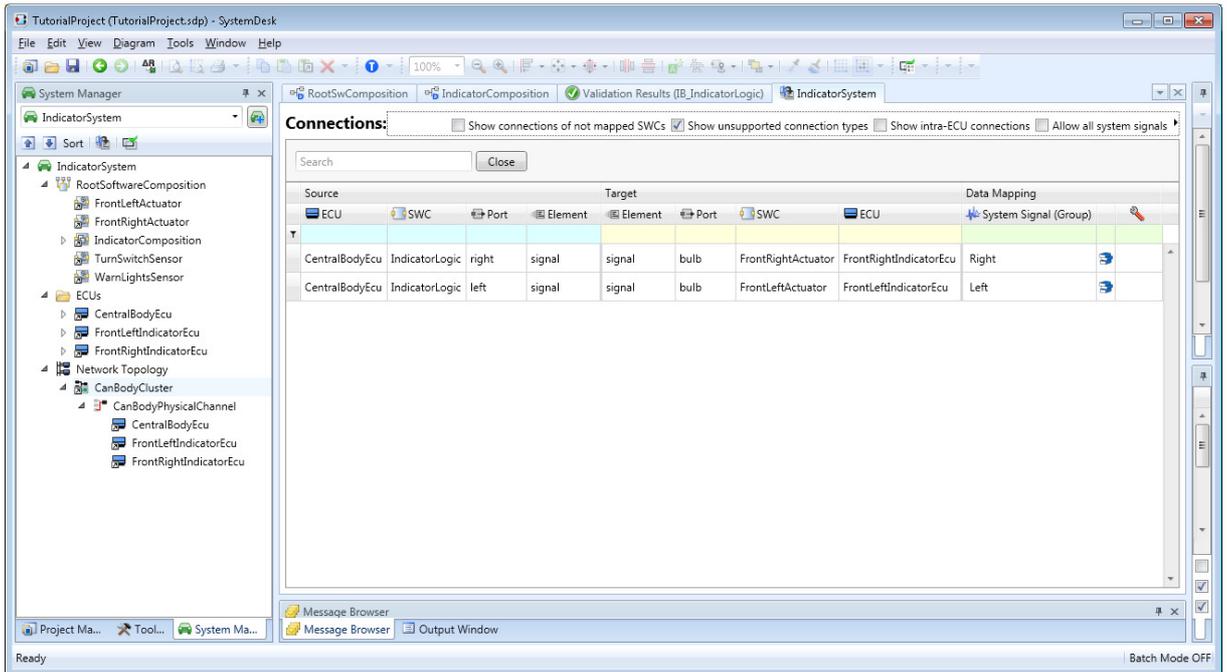


Figure 13: Modeling the System

2.2.4 Step 4: Exporting a system extract

It is recommended to export the architecture as system extract for EB tresos Studio.

You can do this via the context menu of the system. Before you export the architecture, you can choose the ECU(s) which shall be contained in the system extract. Then you can import the resulting file import in EB tresos Studio.

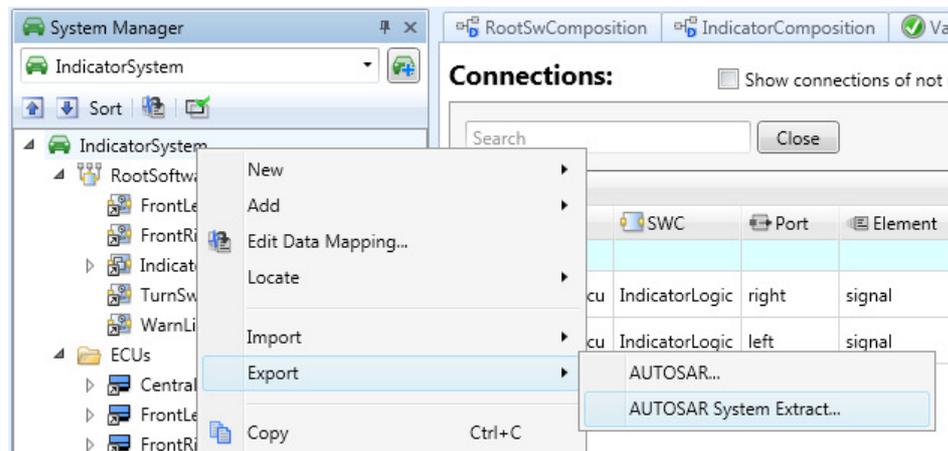


Figure 14: Creating a system extract

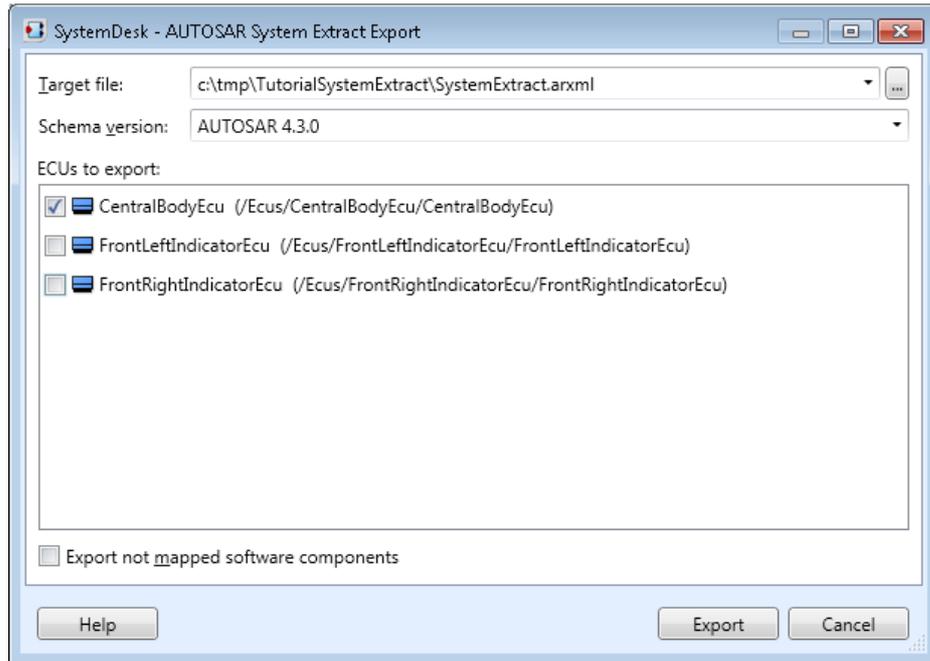


Figure 15: Selecting the ECUs for a system extract

Note:

It is not mandatory to use the system extract for exchange between SystemDesk and EB tresos Studio. The importers in EB tresos Studio are quite flexible and allow the import of arbitrary file structures. It is for example possible to import the same AUTOSAR files that contribute the SystemDesk project into EB tresos Studio directly.

The vast possibilities on how to organize and structure the files of an AUTOSAR project are omitted in this document for the sake of simplicity.

2.3 Configuration and generation of basic software

This section describes those parts of the workflow in detail which you must perform in EB tresos Studio. The so called workflow guides you through this configuration job which includes the following main issues:

- Creating a new ECU configuration project (section 2.3.2)
- Importing the system extract created by SystemDesk (section 2.3.3)
- Mapping of the application events onto Os tasks of the basic software (section 2.3.4.5)
- Generating the basic software and the RTE

2.3.1 Overview

The system extract which has been created with SystemDesk is the base to configure the AUTOSAR basic software (EB tresos AutoCore). It contains information about the software components and their runnables, ports, interfaces and so on.

Following the AUTOSAR methodology this system extract is imported into EB tresos Studio.

The configuration of the EB tresos AutoCore in EB tresos Studio is done in three major steps. These steps are the following, see also Figure 16:

1. Create ECU configuration project in EB tresos Studio
2. Configure AUTOSAR basic software (EB tresos AutoCore Generic)
 - a. Import system extract
 - b. Complete configuration of basic software with support of wizards and assistance dialogs
3. Generate basic software and build target code

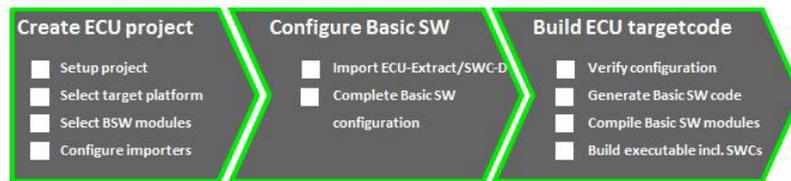


Figure 16: Workflow in EB tresos Studio

The following sections contain a detailed description how to carry out the different steps. To simplify the configuration of the basic software EB tresos Studio provides a special view, the so called **Work-flows** view which contains a step-by-step instruction to guide you through the configuration (see Figure 17).

The **Workflows** view also contains helpful information to each step. For more details on the **Work-flows** view, see the user documentation of EB tresos Studio.

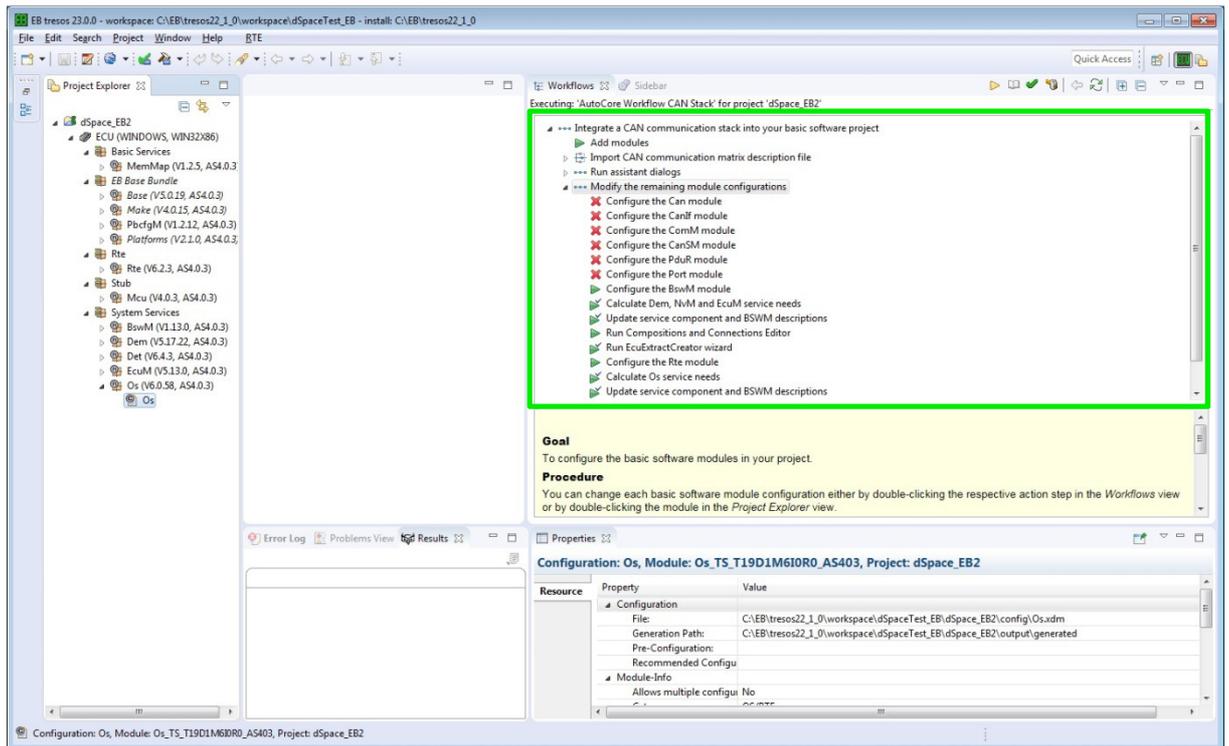


Figure 17: The Workflows view in EB tresos Studio

- Basis Services - MemMap (Memory Mapping)
- EB Base Bundle - Base
- EB Base Bundle - Make
- EB Base Bundle - PbcfgM (Post-build configuration Manager)
- EB Base Bundle - (Platforms)
- Rte - Rte (Runtime Environment))
- Stub - Mcu (Micro controller unit driver; is used as stub in Windows)
- System Services - BswM (Basic Software Mode Manager)
- System Services - Dem (Diagnostics event Manager)
- System Services - Det (Development error tracer)
- System Services - EcuM (Ecu State Manager)
- System Services - OS (Operating System)

The name ECU (WINDOWS, WIN32X86) means the platform of the EB tresos project. In this case real hardware is not used, because WINDOWS is used as platform.

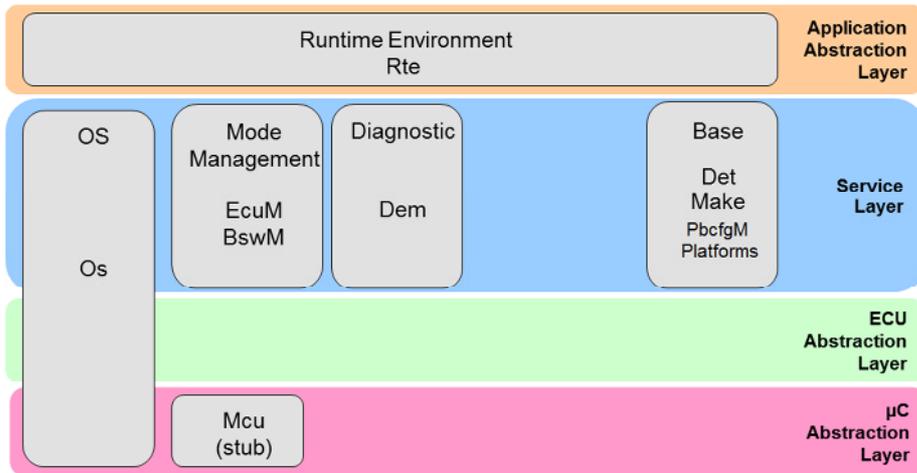


Figure 18: Basic Software modules used in this example

2.3.2 Creating a new ECU configuration project

The first step is the creation of a new ECU configuration project. Follow the instructions within the Workflows view to initially setup your project. The instructions are shown in the section Create new project (*from template*), see Figure 19.

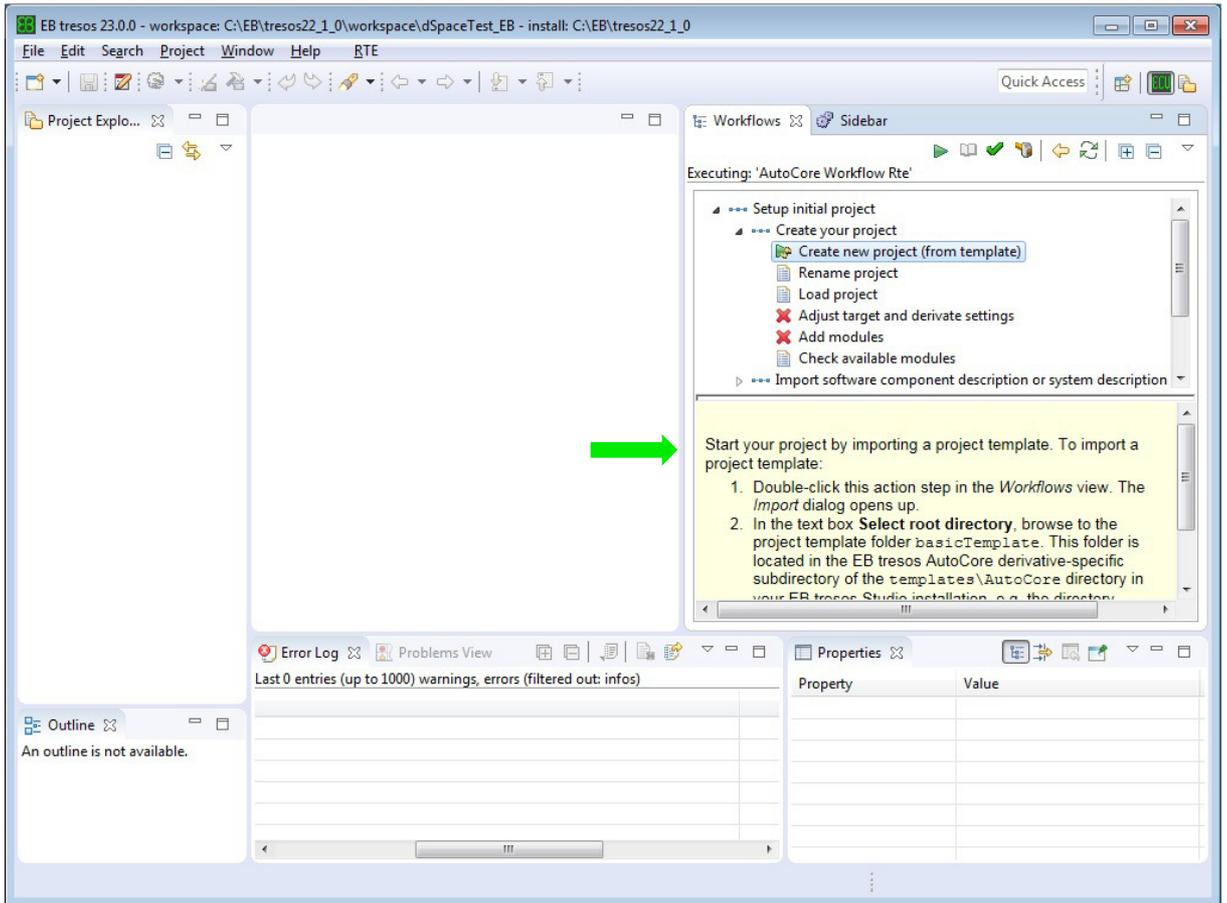


Figure 19: Instructions to create an ECU configuration project

You can find the basic template project which shall be imported in the EB tresos Studio installation directory, see Figure 20.

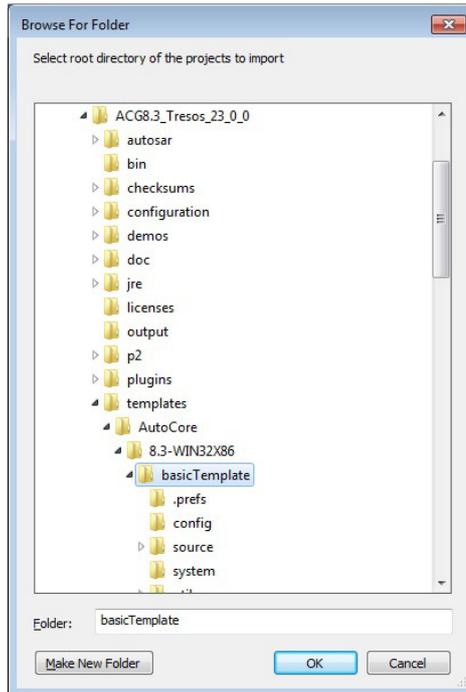


Figure 20: Location of basic template project in the EB tresos Studio installation directory

The completely filled out dialog is displayed in Figure 21. Make sure to select the option **Copy projects into workspace** for the project import.

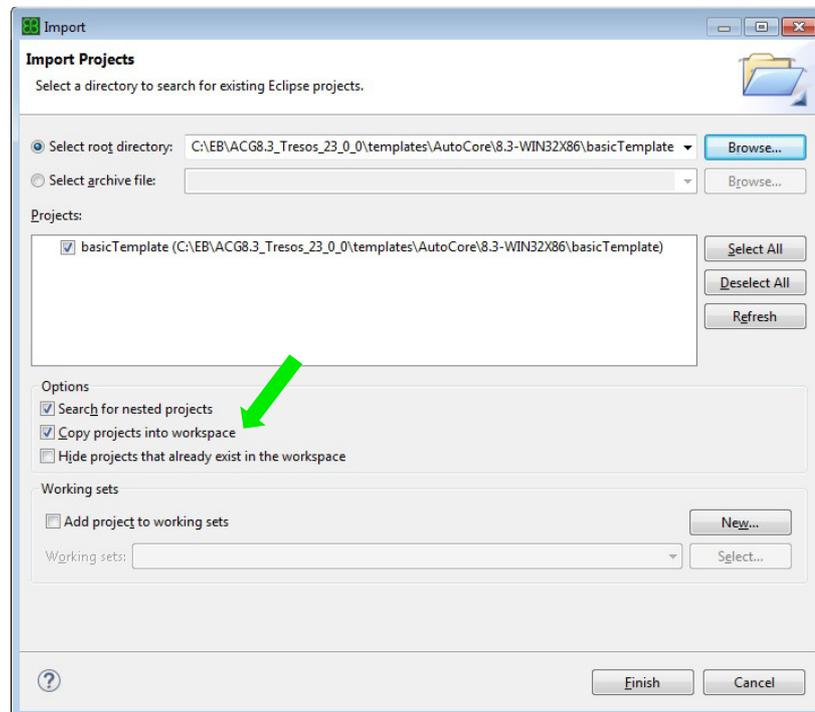


Figure 21: Dialog to import a project in EB tresos Studio

After you complete the working step Create a new project (*from template*), the project appears as displayed in Figure 22. To load the project, double-click **ECU (WINDOWS, WIN32X86)**.

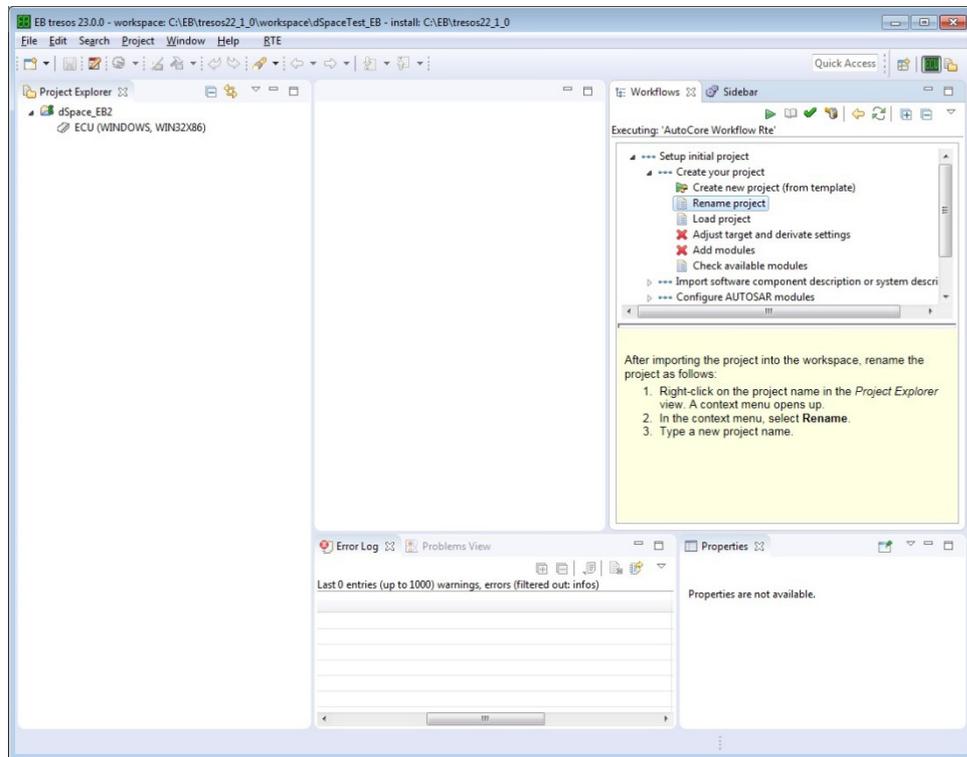


Figure 22: The Project Explorer view shows the new project before loading

After the project is loaded, a plus button appears before **ECU (WINDOWS, WIN32X86)**. To open the tree, click the plus button. The project tree looks like the next Figure 23.

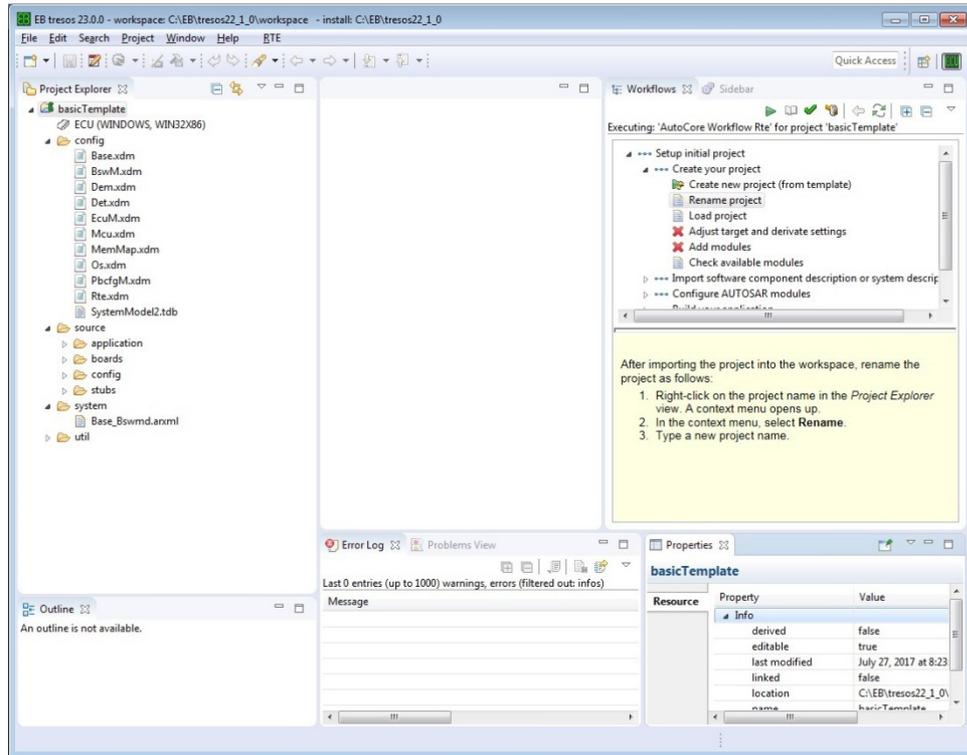


Figure 23: The Project Explorer view shows the new project after loading

2.3.2.1 Renaming the project

The renaming is described in the light yellow subsection of Figure 23. The steps are also shown in the following Figure 24.

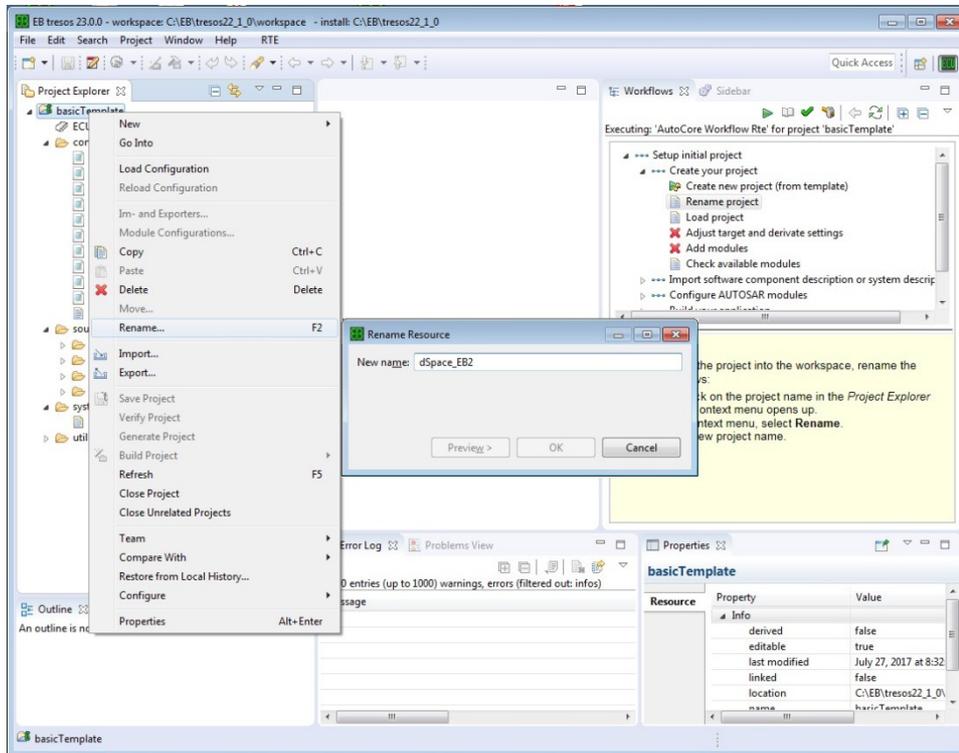


Figure 24: The Project Explorer view shows the new project

2.3.3 Importing the system extract

The configuration of the basic software starts with the import of the system extract which has been created with SystemDesk, see section 2.2. To do the import, you must configure an AUTOSAR system description importer in EB tresos Studio. Figure 25 shows the necessary steps to create it. The green arrow shows the button to open the **New Importer/Exporter** dialog.

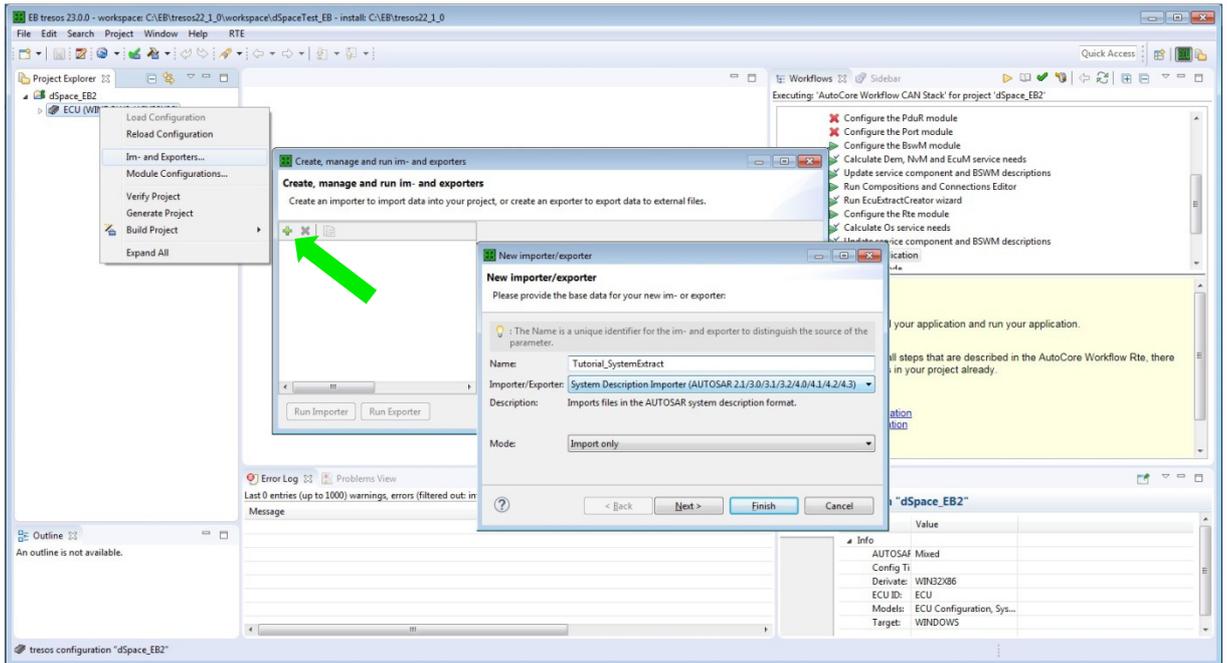


Figure 25: Instructions to import the system extract into the project

Provide the name for the importer.

In Figure 26 the name is chosen equally to the file name of the import file.

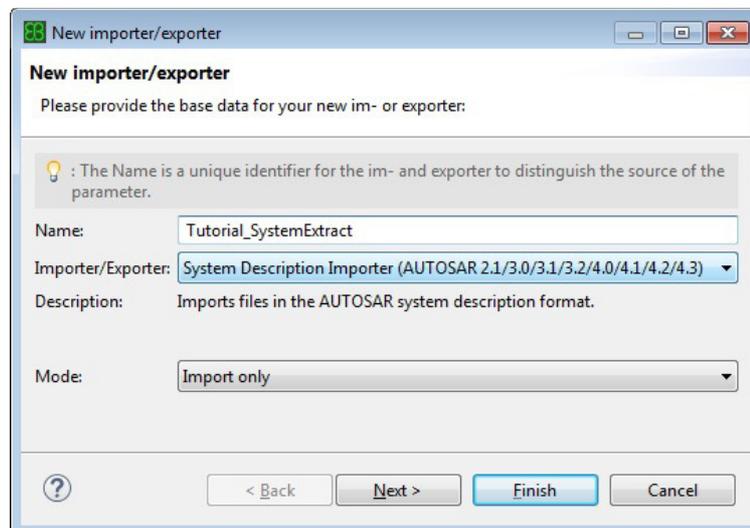


Figure 26: Importer configuration for the import of the system extract

In the configuration tab **All Models** the location of the input file must be specified. Here it is the sys-tem extract which was exported out of SystemDesk. It is recommended to first copy this into the project folder in the workspace because this file is always project-specific. Switch the **AUTOSAR schema variant** to standard and then select the system and an ECU instance, see Figure 27.

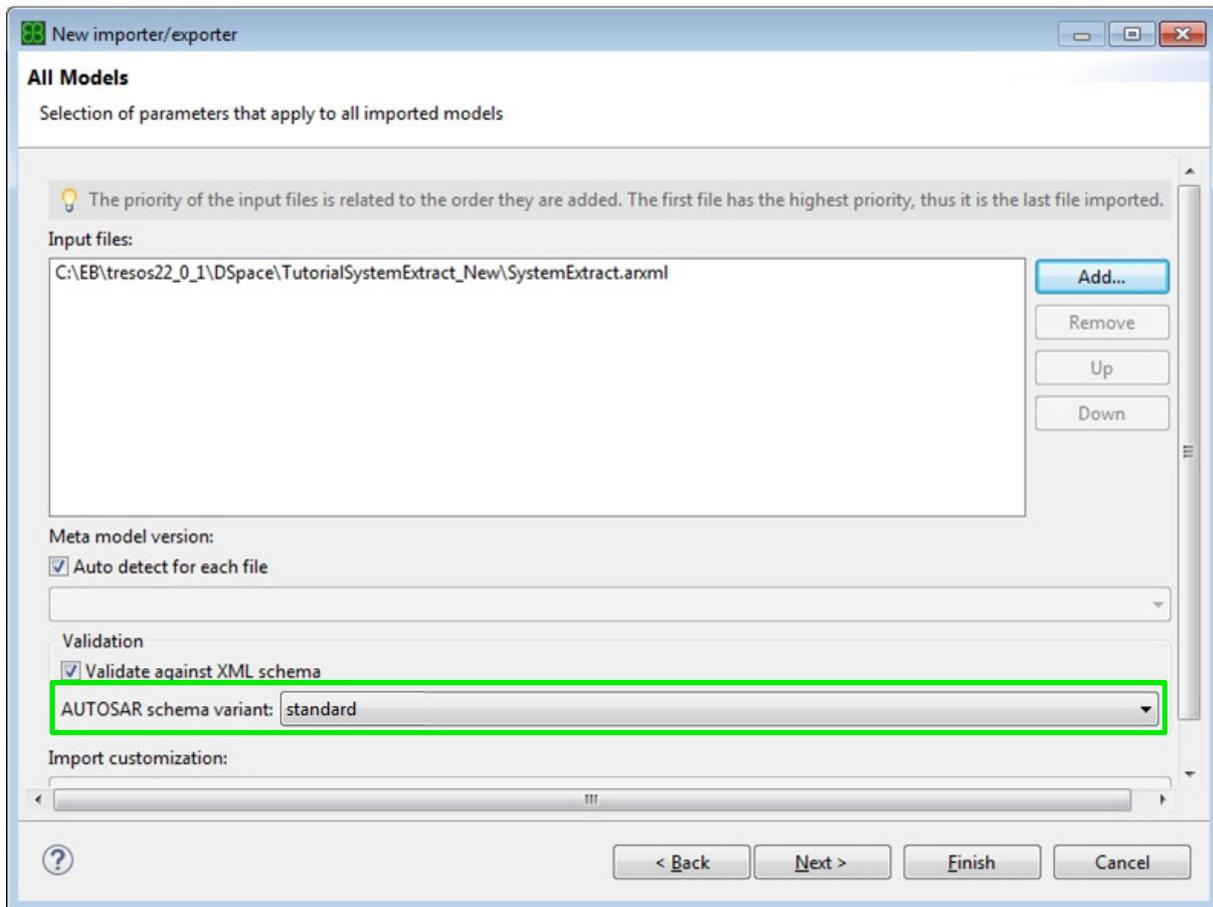


Figure 27: Importer configuration: selection of system and ECU instance

Click the **Next >** button to open the **System Model Import** dialog.
Here the import options must be set according to Figure 28.

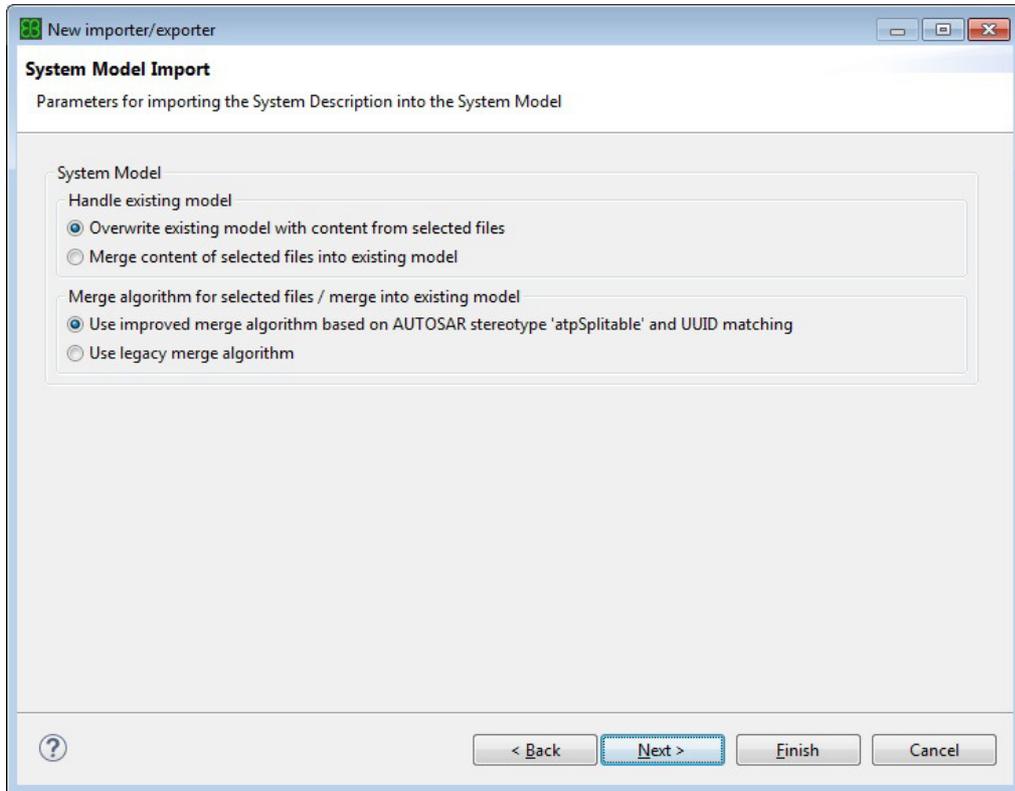


Figure 28: Importer configuration: system model settings

Click the **Finish** button, to close the dialog box.
The **Create, manage and run im- and exporters** dialog opens.
You can see all options and settings you made in this dialog, see Figure 29.

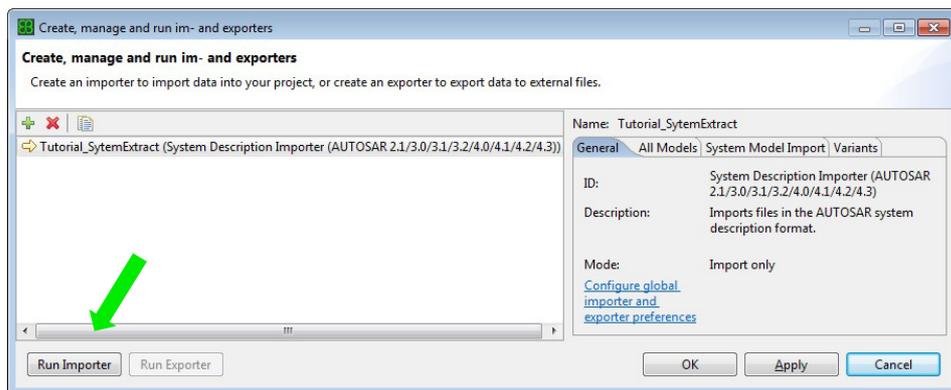


Figure 29: Importer configuration: Create, manage and run im- and exporters

Click the **Run Importer** button in this dialog.
Now the importer starts and imports the system extract into the internal EB tresos Studio format.

2.3.3.1 Module Configuration Check

The **Error! Reference source not found.** shows all modules which are needed for the next steps.

Make sure to select all modules as shown in the figure.

Select any missing module from the **Available Modules** tree and click the plus button in order to add it to the active list of **Module Configurations**.

The steps may be necessary for the modules MemMap and PbcfgM.

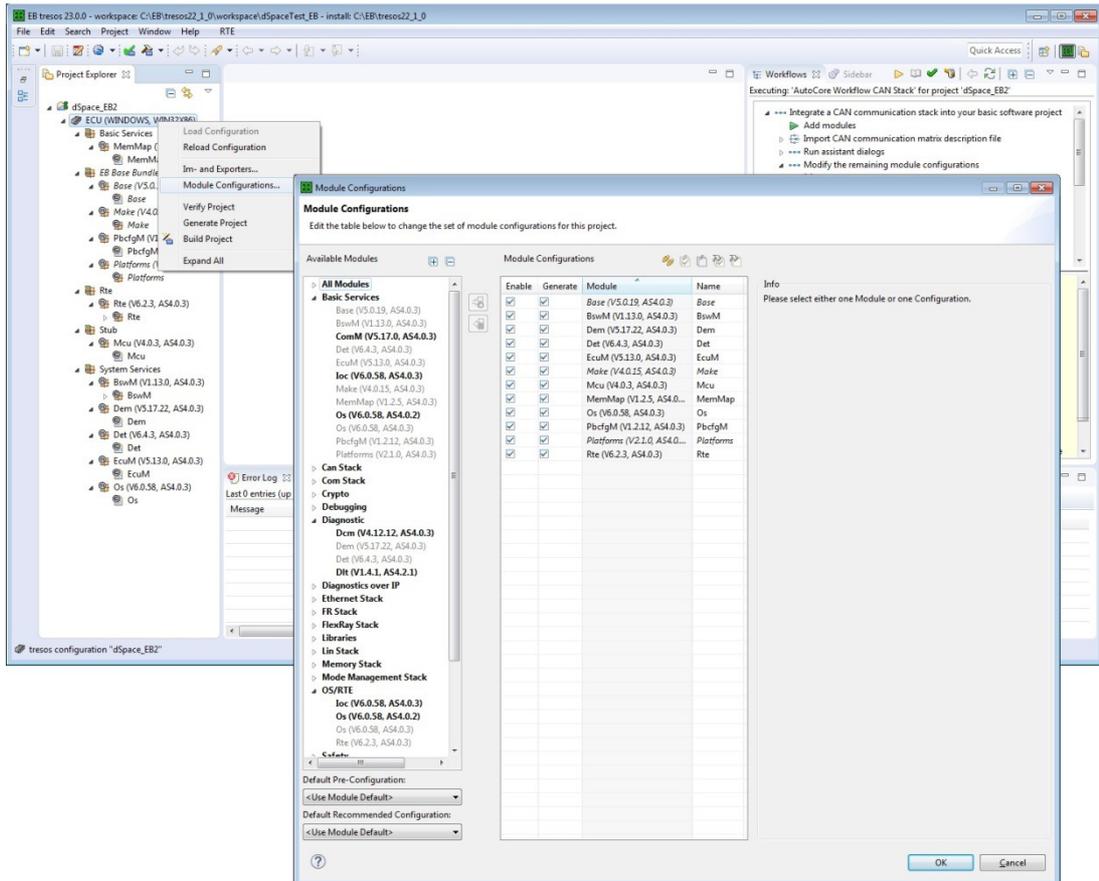


Figure 30: Necessary modules

2.3.4 Configuration of AUTOSAR modules

The configuration of the AUTOSAR basic software modules applies for the modules which are listed in section 2.3.3.1. The **Workflows** view displays the necessary steps to be executed, see Figure 31.

For each configuration step the help view below (Help subsection) the instruction list provides useful information.

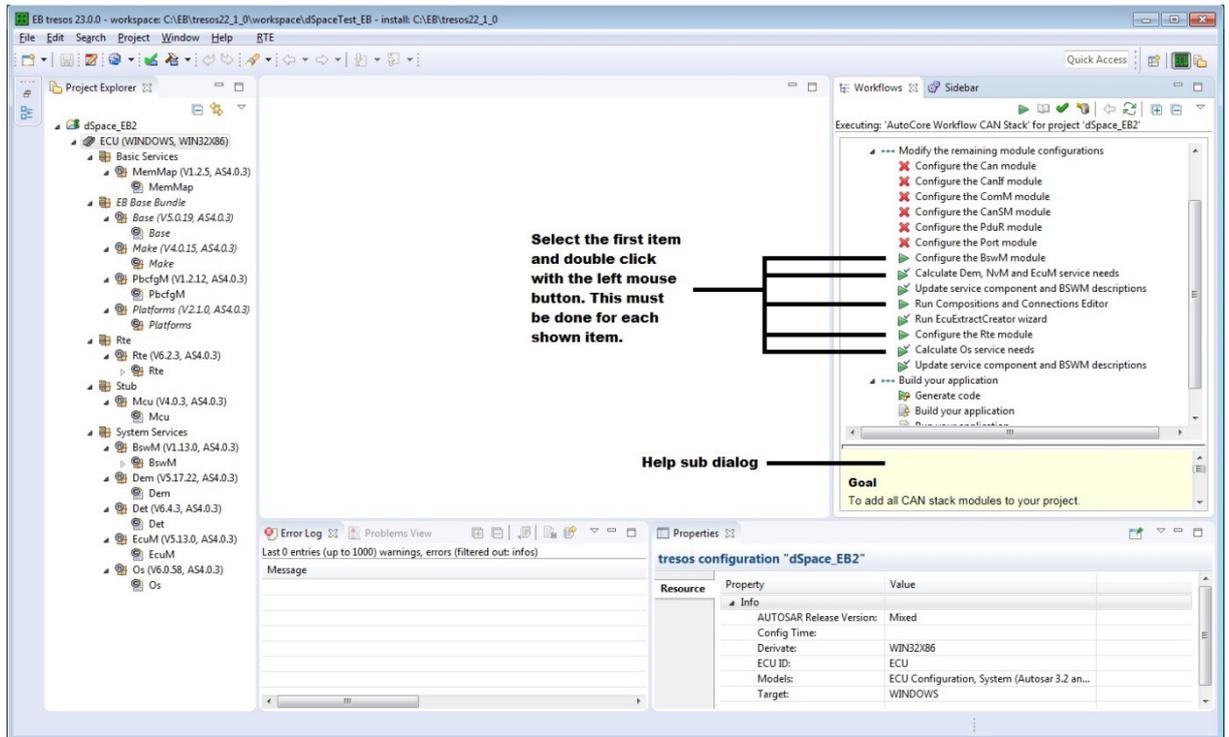


Figure 31: Instructions to configure the basic software modules

When you select Run Compositions and Connections Editor you must select the last point inside the dropdown list box.

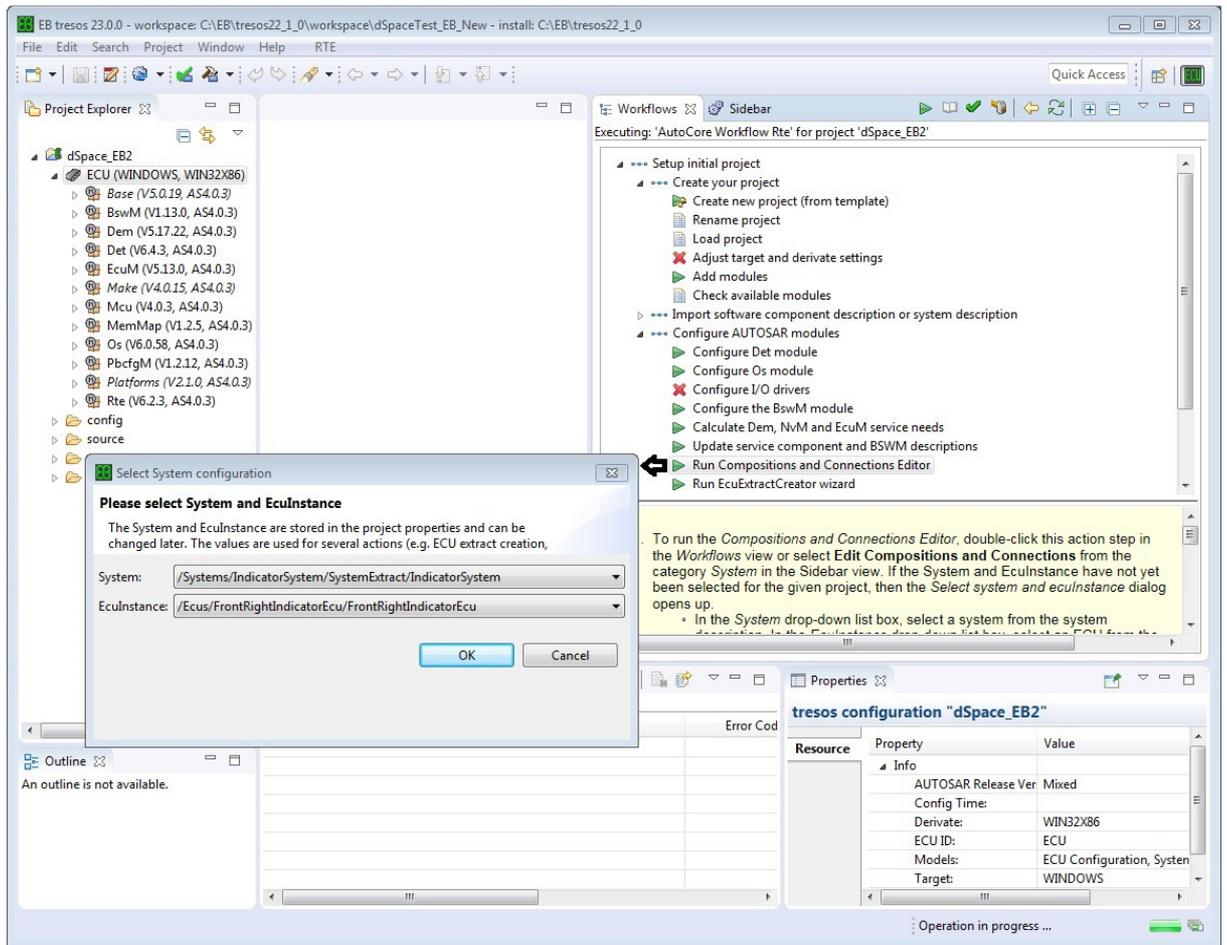


Figure 32: Activation points in “Run Compositions and Editor”

Hint:

Close any document window in the middle section if no longer needed.

2.3.4.1 Configuration of the Os module

The system extract contains software components and their runnable entities.

These runnable entities must be mapped in the Os module.

2.3.4.1.1 OsTast

You must map the runnable entities to Os tasks.
The tasks must be created in the Os configuration.

Double Click the action step Configure Os module in the **Workflows** view to open the editor for the Os configuration.

Now select the **OsTask** tab and add the following tasks via the + button, see Figure 33.

- RteTime_Task
- RteEvent_Task
- RteCat2_Task

For more information, see the help section within the **Workflows** view or the EB tresos AutoCore documentation.

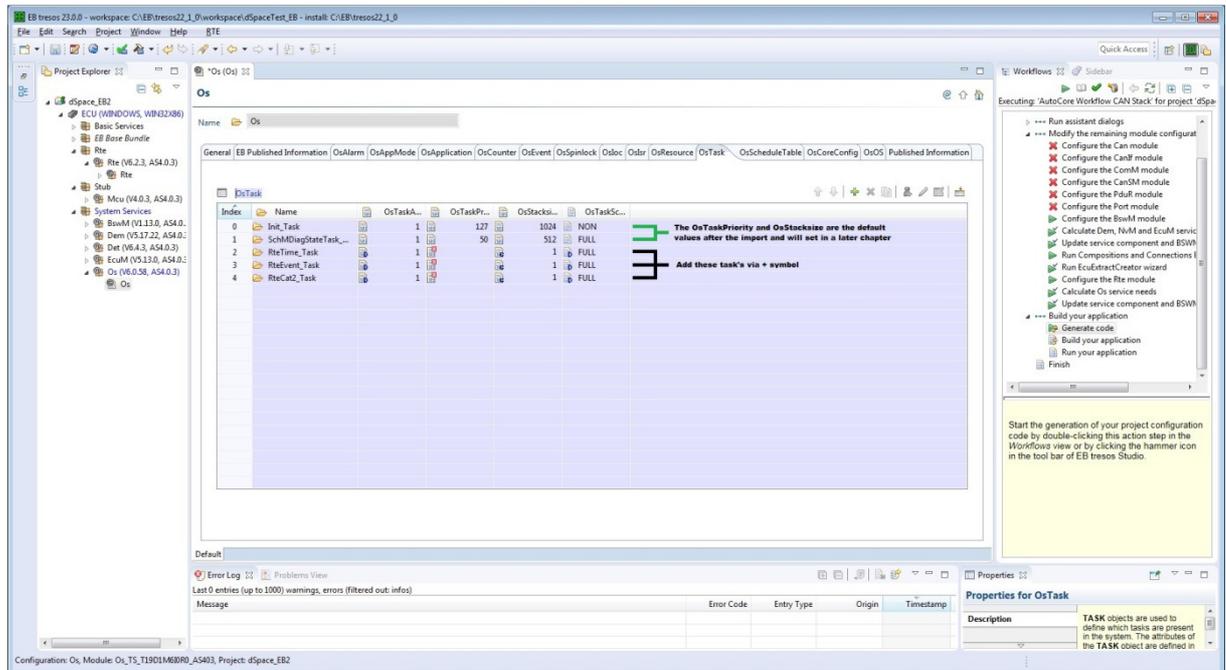


Figure 33: New tasks within the Os task configuration

2.3.4.1.2 OsScheduleTable

Select the **OsScheduleTable** tab.

Click the + button and set the name field to *Ret_DefaultScheduleTable*.

Double Click the entry *Ret_DefaultScheduleTable*, see Figure 34

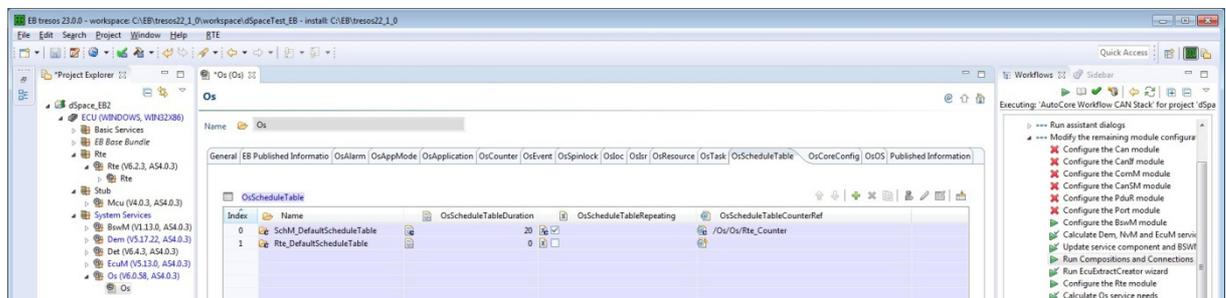


Figure 34: Os schedule table configuration

Select the **General** tab and change the shown entries *OsScheduleTableDuration*, *OsScheduleTableRepeating*, *OsScheduleTableCounterRef*, according to Figure 35.

Warning:

Exactly these modifications must be applied to the entry “*SchM_DefaultScheduleTable*” also.

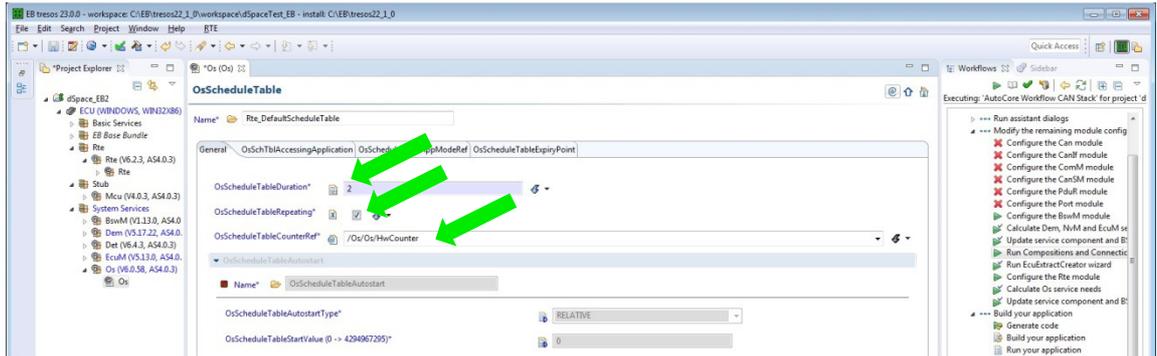


Figure 35: Os schedule table configuration (submenu_1, tab General)

Select the **OsScheduleTableExpiryPoint** tab and double-click the “**EP_0**” entry, see Figure 36.

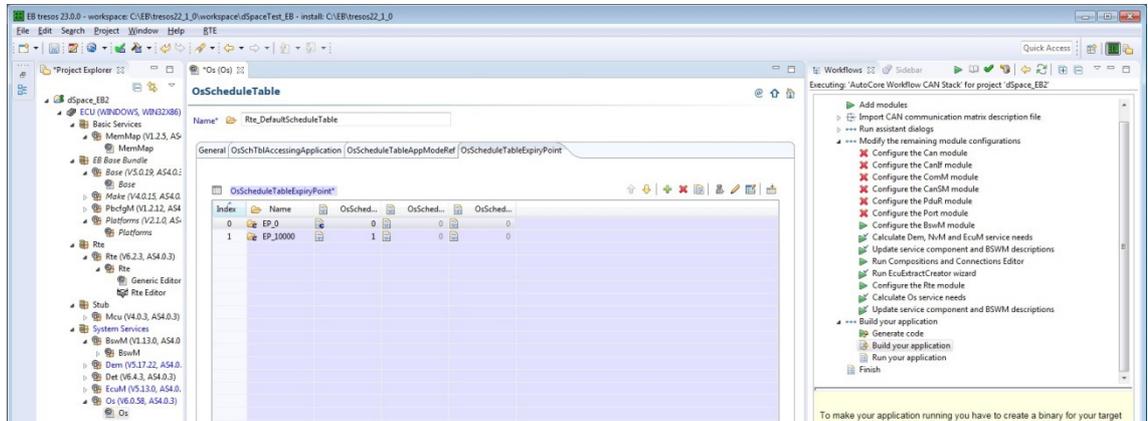


Figure 36: Os schedule table configuration (submenu_1, tab OsScheduleTableExpiryPoint)

Select *EP_0* in Figure 36 and double-click this entry.

In the next menu you must enter the shown task names, see Figure 37.

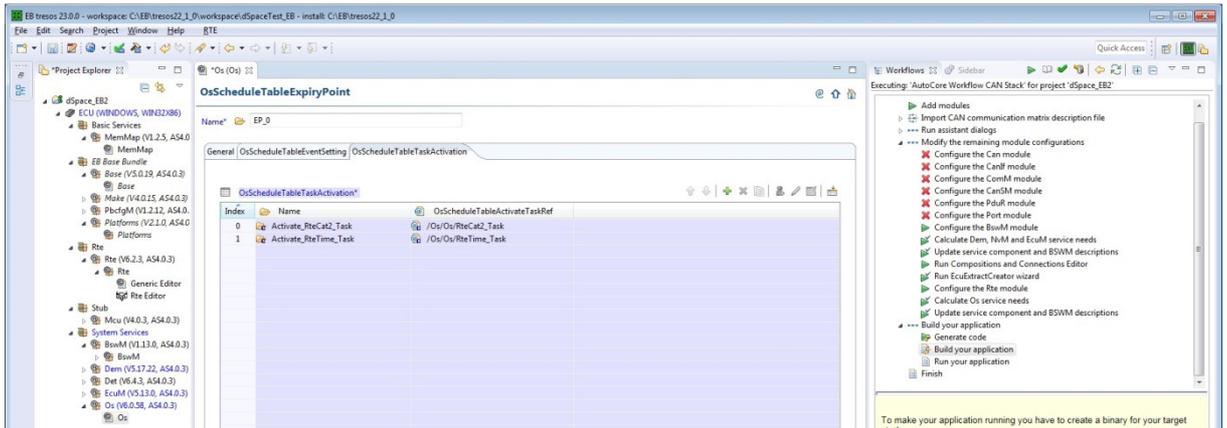


Figure 37: Os schedule table configuration (submenu_1 -> submenu_2 for EP_0)

Select *EP_10000* in Figure 36 and double-click this entry.

In the next menu you must enter the shown task name, see Figure 38.

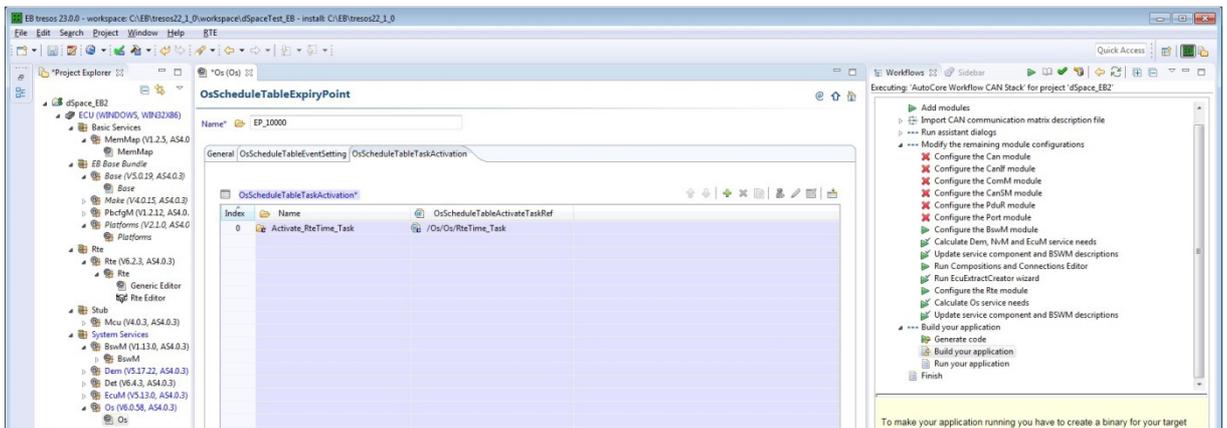


Figure 38: Os schedule table configuration (submenu_1 -> submenu_2 for EP_10000)

2.3.4.1.3 OsEvent

Select the **OsEvent** tab and add via the + button the shown entries of Figure 39.

Hint:

If the names from Figure 39 do not exist, click **Calculate Dem, NvM and EcuM service needs** again. If the names are still not inserted then proceed with section 2.3.4.5 and repeat this step later.

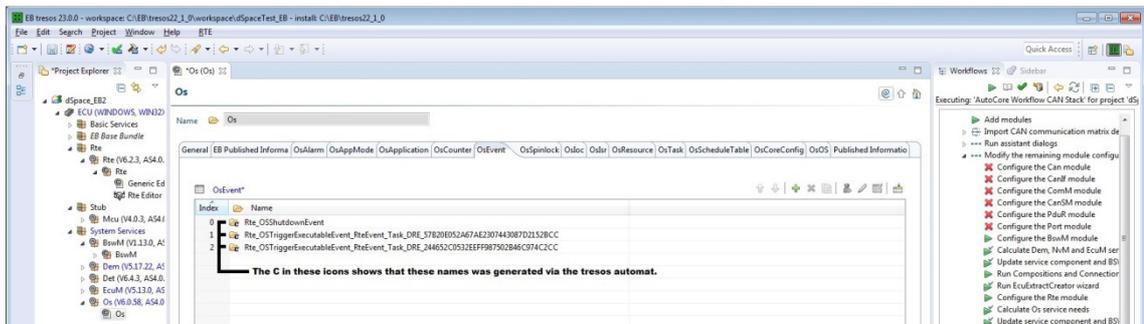


Figure 39: Os event table configuration

2.3.4.1.4 OsOs

Select the **OsOs** tab and disable the switch **OsProtectionHook** shown in Figure 40.

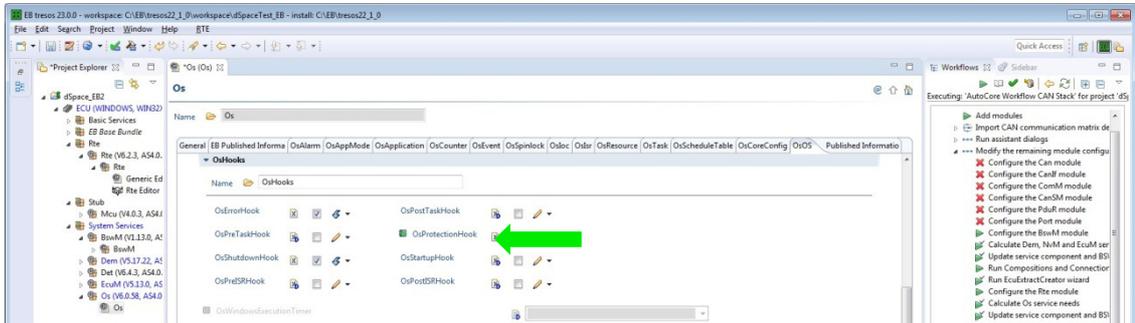


Figure 40: Os os switch configuration

Save the OS modification and close the OS configuration.
Reopen the OS, see section 2.3.4.1.1 and select the **OsTask** tab.
Make sure to modify the values according to Figure 41.

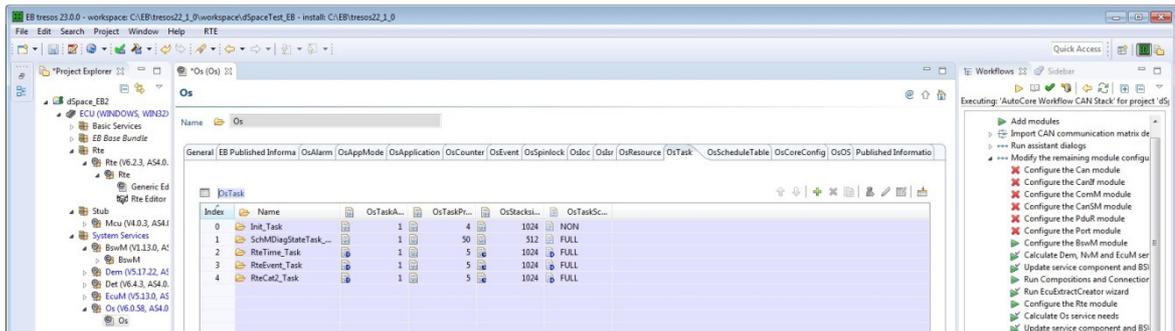


Figure 41: Os task configuration

Save the OS modification and close the OS configuration.

2.3.4.2 Configuration of the BswM module

Double-click Configure the BSwM module according to Figure 31.

2.3.4.3 Calculate Dem, NvM and EcuM service needs

Hint:

The basic software modules are not independent of each other. These dependencies are resolved by the Service Needs Calculator which for example automatically adds the init functions of the different basic software modules to the EcuM (EcuM initializes the ECU) or adds the required Dem events to the Dem configuration. To run the Service Needs Calculator double-click the action step in the **Work-flows** view. The service needs are now calculated and the configurations of the affected modules are automatically updated.

2.3.4.4 Update service component and BSWM descriptions

Double-click **Update service component and BSWM descriptions** according to Figure 31 .

Hint:

The *Service Component and BSWM Description Updater* generates and imports the basic software module description (BSWMD) of all modules.

The update of the BSWMD is a prerequisite for the configuration of the main-function-to-task mapping and of the configuration of the BSW exclusive area in the *RTE*. Moreover, the *Service Component and BSWM Description Updater* also generates and imports the service component descriptions for service modules, e.g. *Det*. After this step, the service components are available and can be connected to the application software components.

2.3.4.5 Configuration of the Rte module

2.3.4.5.1 Configuration of the Rte via Generic Editor

The configuration of the RTE begins when you add new modules to the BSW Module Instances (BSW_Base, BSW_PbcfgM) as shown in the **Error! Reference source not found..**

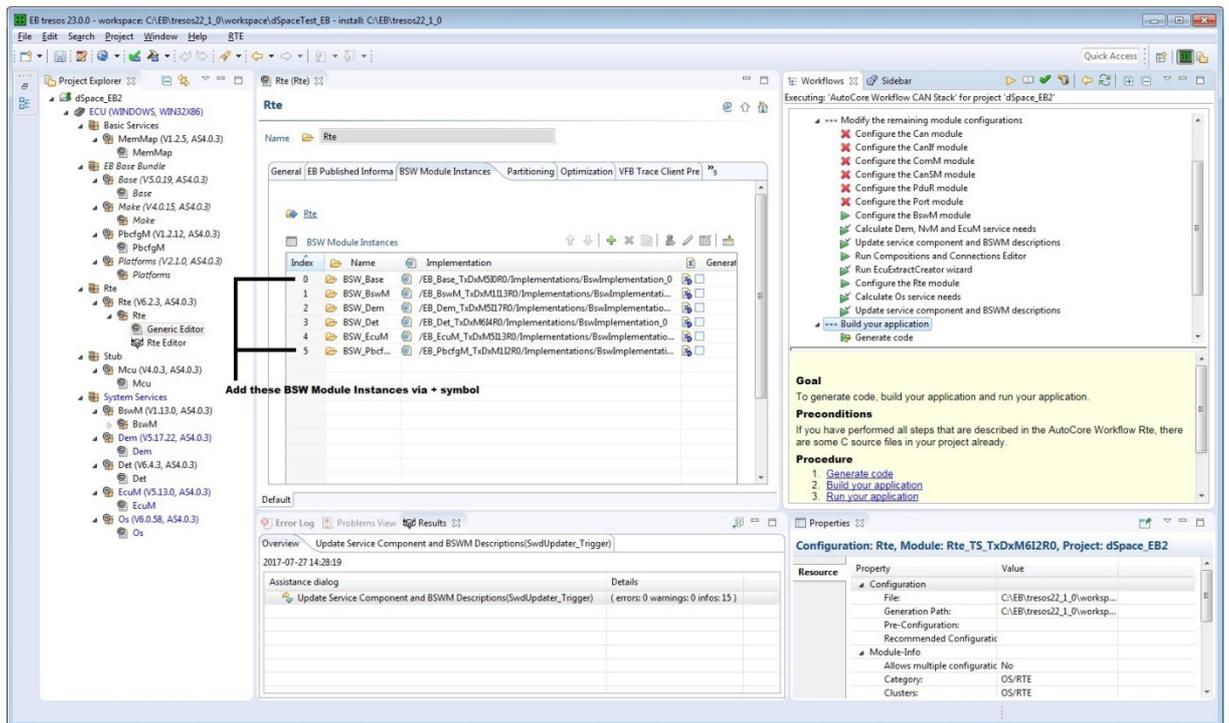


Figure 42: Rte Generic Editor

Set the **Rte Generation Output** field to FULL, see Figure 43.

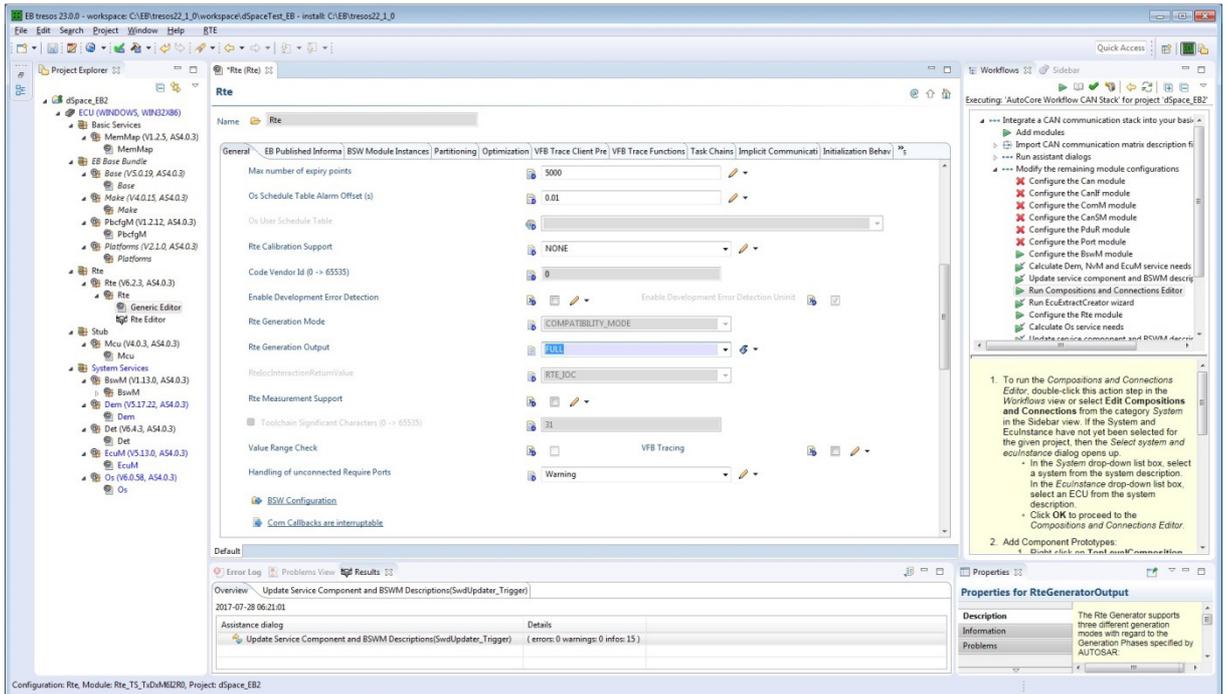


Figure 43: Rte Generic Editor General tab

2.3.4.5.2 Configuring the Rte via Rte Editor

The configuration of the Rte is done with a special RTE Editor. To launch this editor double-click the action step *Configure Rte* module within the workflow and then select **Rte Editor**, see Figure 44.

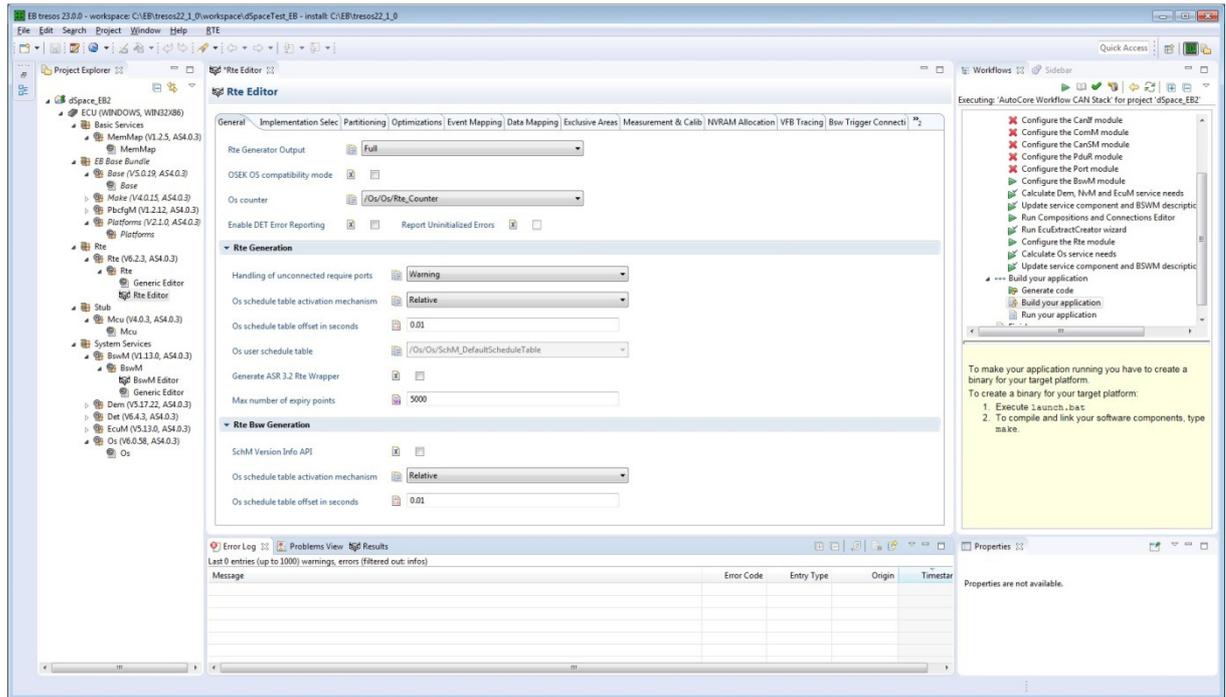


Figure 44: Rte Generator output settings and Os counter settings in the Rte Editor

To generate RTE code as well as the BSW scheduler code the item Rte Generator Output in the **General** tab must be set to **Full**.

The RTE needs a reference to an Os counter.

The basic template which is used in this example already provides the software counter Rte_Counter.

Verify that this counter is selected in the **General** tab of the Rte Editor, see Figure 44.

Now the RTE events must be mapped to the Os tasks.

Open the tab **Event Mapping** in the Rte Editor to do the mapping. In the upper table the events, their executable entities and SWC instances are listed, see Figure 45.

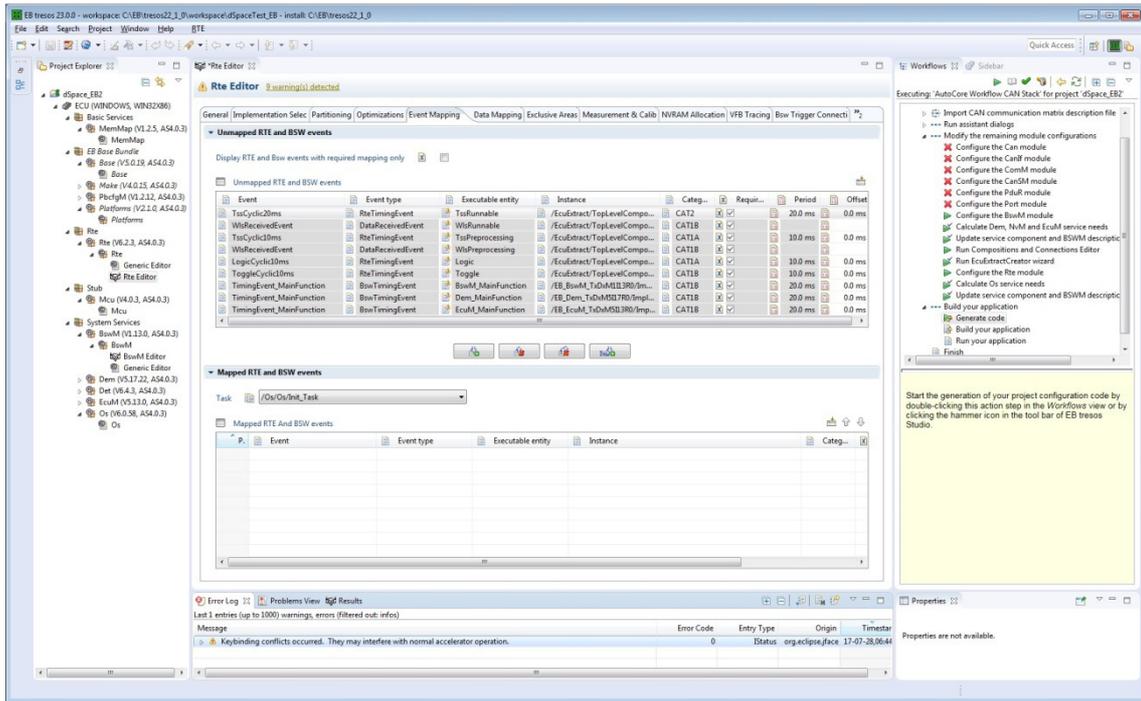


Figure 45: List of events that must be mapped

In the middle of this tab you can select a task on which events may be mapped. The lower table lists the events that are mapped to the selected task.

To map, select the right event to the corresponding task, see Figure 45.

After that select the events of the upper table and click the **Map** button (button with + symbol). The selected events are now mapped in Figure 46.

▼ Mapped RTE and BSW events

Task

Mapped RTE And BSW events

P.	Event	Event type	Executable entity	Instance	Category	Require
1	TimingEvent_MainFunction	BswTimingEvent	BswM_MainFunction	/EB_BswM_TxDxM113R0/Imple...	CAT1B	<input checked="" type="checkbox"/>
2	TimingEvent_MainFunction	BswTimingEvent	Dem_MainFunction	/EB_Dem_TxDxM5117R0/Imple...	CAT1B	<input checked="" type="checkbox"/>
3	TimingEvent_MainFunction	BswTimingEvent	EcuM_MainFunction	/EB_EcuM_TxDxM5113R0/Imple...	CAT1B	<input checked="" type="checkbox"/>

▼ Mapped RTE and BSW events

Task

Mapped RTE And BSW events

P.	Event	Event type	Executable entity	Instance	Category	Require
1	LogicCyclic10ms	RteTimingEvent	Logic	/EcuExtract/TopLevelCompositi...	CAT1A	<input checked="" type="checkbox"/>
2	ToggleCyclic10ms	RteTimingEvent	Toggle	/EcuExtract/TopLevelCompositi...	CAT1B	<input checked="" type="checkbox"/>
3	TssCyclic20ms	RteTimingEvent	TssRunnable	/EcuExtract/TopLevelCompositi...	CAT2	<input checked="" type="checkbox"/>

▼ Mapped RTE and BSW events

Task

Mapped RTE And BSW events

P.	Event	Event type	Executable entity	Instance	Category	Require
1	WisReceivedEvent	DataReceivedEvent	WisRunnable	/EcuExtract/TopLevelCompositi...	CAT1B	<input checked="" type="checkbox"/>
2	WisReceivedEvent	DataReceivedEvent	WisPreprocessing	/EcuExtract/TopLevelCompositi...	CAT1B	<input checked="" type="checkbox"/>

▼ Mapped RTE and BSW events

Task

Mapped RTE And BSW events

P.	Event	Event type	Executable entity	Instance	Category	Require
1	TssCyclic10ms	RteTimingEvent	TssPreprocessing	/EcuExtract/TopLevelCompositi...	CAT1A	<input checked="" type="checkbox"/>

Figure 46: Mapping of events to a task

Close the Rte Editor.

2.3.4.6 Calculating Os and Com Service Needs

It is recommended that you run the *Service Needs Calculator* wizard again to add the Os and Com objects required by the *RTE* module to the Os configuration. Since this requires a consistent *RTE* configuration, this step is recommended after you have completed the configuration of the *RTE*. Double-click the action step *Calculate Os and Com service needs*.

2.3.4.7 Updating service component and BSWM descriptions

The last steps have changed the ECU configuration, on which the BSWMD depends. Thus, it is recommended to update the basic software module description again.

Double-click the action step *Update service component and BSWM descriptions*.

2.3.5 Verifying the configuration

The configuration of the basic software modules is now complete.

Click the **Verify selected project** button, to verify the configuration and check for configuration errors, see Figure 47.

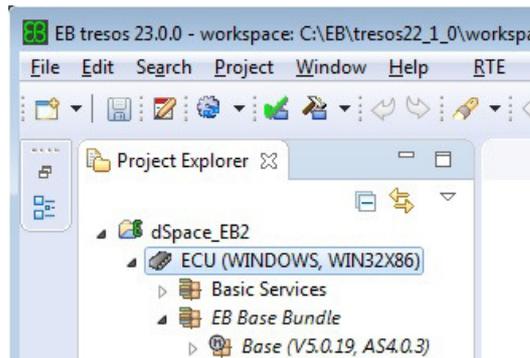


Figure 47: Verify the configuration

After successful verification the code can be generated for the basic software modules. To do so, click the **Generate** button in the tool bar, see Figure 48.

You can find the generated code in the folder:

```
<tresos>/workspace/<your-tresos-project>/output/generated
```

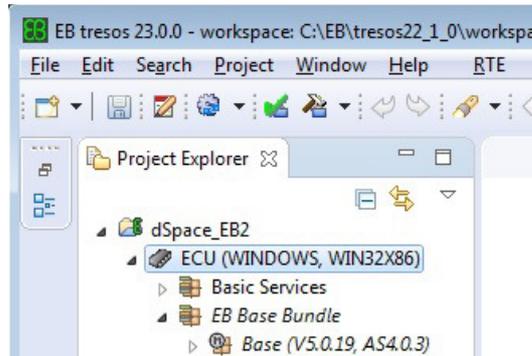


Figure 48: Generate code for the basic software modules

2.3.6 Adding the communication parts to the basic software

Switch to the workflow *CAN stack* or *FlexRay stack*, to add the communication modules and to import the communication data, as shown in Figure 49. Then follow the workflow instructions for CAN or FlexRay in the same way as in the previous chapters for the workflow *Setup an initial project*.

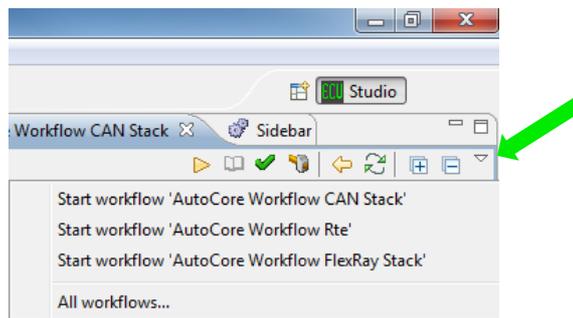


Figure 49: Workflows for communication

2.3.7 Export of the service component interfaces

As described in section 2.2.2.1 the software component description of the basic software can be generated, see Figure 50.

Click the generate code button.

Chose generate_swcd

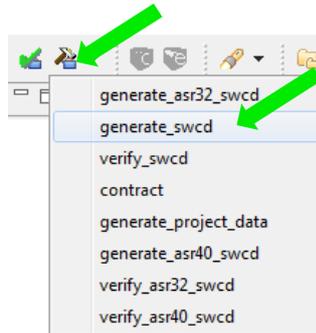


Figure 50: Generate SW-CD of service modules

You can find the output files in the folder:

`<tresos>/workspace/<your-tresos-project>/output/generated/swcd/`

You can find the interfaces in the file(s) `<module>_swc_interface.arxml`.

In the current example, no basic software services are configured. Therefore these files are not generated.

To generate a SWCD of a basic software module, you must configure this in EB tresos Studio. Especially the parameter **Enable Rte Usage** must be enabled to switch on the port generation. The generation of the SWCD is even possible if there are errors in the configuration project.

The generated interfaces are imported into SystemDesk to ensure that the application SWC uses interfaces which are compatible to the basic software.

So the overall workflow is as follows:

- Preconfigure the basic software in EB tresos Studio and generate the SWCD of the basic software services.
- Import these service interfaces into SystemDesk and create a complete system description.
- Import the system description into EB tresos Studio and finalize the configuration of the RTE and the basic software.

2.3.8 Quick test building of generated software

If you want to make a quick test you can do this without the following chapters. Files that contain ex-ample implementations of the application software components are provided along with this applica-tion note. The resulting executable will print status messages to the command line.

If you want to do this, use the documentation in the appended `dSpace_TargetLink_Tresos` zip-file and follow the description in the `ReadMe.txt`.

3 Function development

3.1 Overview

This chapter describes how to integrate the function development in the workflow described in chapter 2.

If you use hand-programmed code, you need to make sure that it fits to the internal behavior described in the software architecture, and that it is compliant to AUTOSAR. The code files are then handed over to be integrated in the EB tresos AutoCore build environment.

If you use TargetLink for code generation, you can use the architecture which is defined in SystemDesk. Figure 51 shows an overview of the interaction of TargetLink with SystemDesk and EB tresos Studio. It extends the workflow described in chapter 2.

Export a software component from SystemDesk as ARXML (or SWC container) and import it in a TargetLink data dictionary. You can then reuse the data types, interfaces, ports, data accesses, runnables etc. for your behavior model. You can also use TargetLink to extend the behavior. TargetLink offers for example a simple possibility to generate implementation data types from application data types, and it creates a constant specification mapping when needed. When the behavior model is complete, you can generate the code for the software component.

The updated AUTOSAR software component descriptions are imported back to SystemDesk via SWC container. New ports conveniently can be connected to other components in SystemDesk. Make sure to validate the project and fix potential errors. The revised system then is exported as SystemExtract and provided to EB tresos Studio.

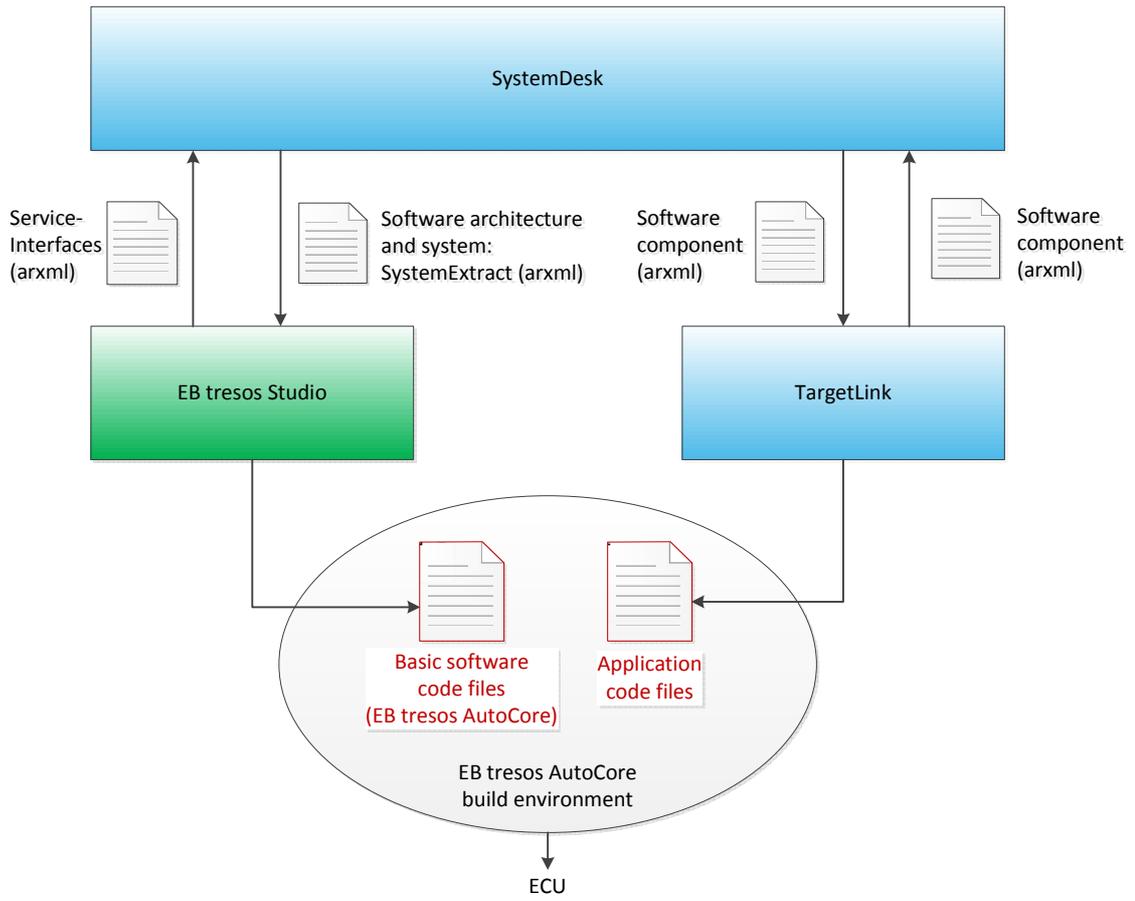


Figure 51: Overview of the interfaces between TargetLink, SystemDesk, and EB tresos Studio

In the following, the single steps are described in more detail. You can find detailed information about the data exchange between SystemDesk and TargetLink in the *Container Management Document*.

3.2 Exporting a software component from SystemDesk

The first step is to export the software component from SystemDesk for which the internal behavior shall be modeled in TargetLink. Before you do this, it is advisable to use SystemDesk's validation: SystemDesk supports the complete AUTOSAR schema. TargetLink, on the other hand, has some limitations regarding specific AUTOSAR features. Therefore, before you export software components from SystemDesk, validate your project for compatibility with TargetLink. This will help to eliminate many possible pitfalls. To use the TargetLink validation, select the applicable validation rule set in the tool bar and then validate your project.

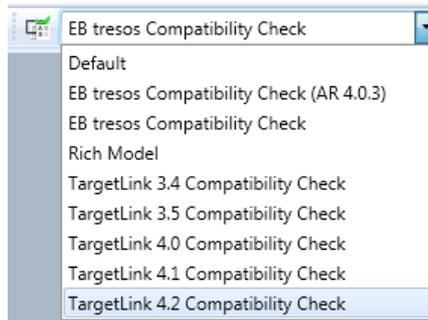


Figure 52: Selecting a TargetLink validation rule set

Then you can export from SystemDesk. TargetLink focusses on single, atomic software components.

Export each SWC from SystemDesk separately. The process of import and export is simplified by the SWC container management functionality, which bundles all necessary files and synchronizes them according to a specified workflow.

To export a software component as container from SystemDesk, you first need to specify which element is exported to which ARXML file in the Container File Explorer.

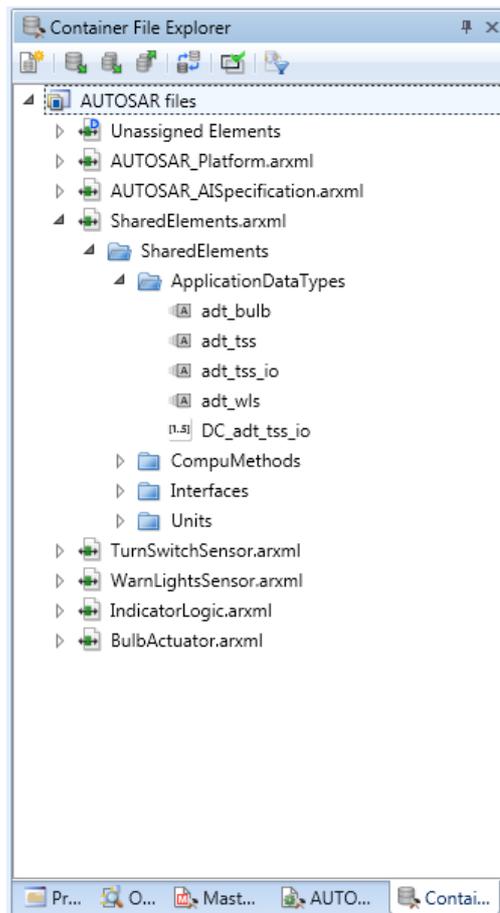


Figure 53: The Container File Explorer

You can export the software component via its context menu: choose **Container Management / Export Container**. A dialog shows which ARXML files are added to the container. After a confirmation, the export is performed.

You can find more details about file assignment and the container manager in the *SystemDesk* Guide and the *Container Management Document*.

3.3 Importing a software component in TargetLink

To import the software component in TargetLink, open the AUTOSAR data dictionary in which you want to import the software component.

If you create a new data dictionary, make sure that you create it according to the AUTOSAR template.

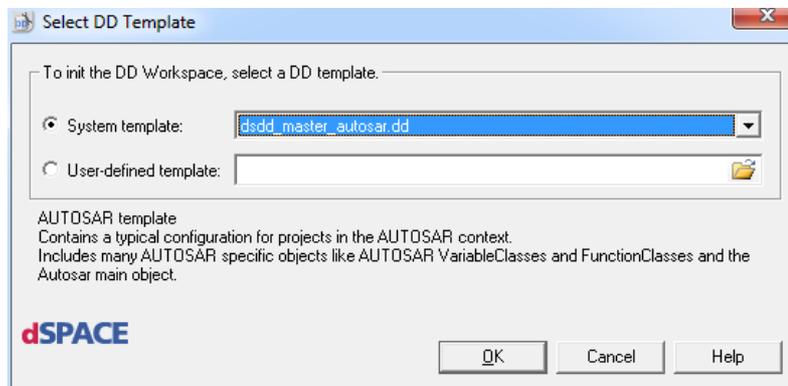


Figure 54: Creating a new data dictionary according to the AUTOSAR template

When you opened the data dictionary, select **File / Import / From Container** and select the container with the software component you exported from SystemDesk. Confirm the settings and TargetLink imports the software component.

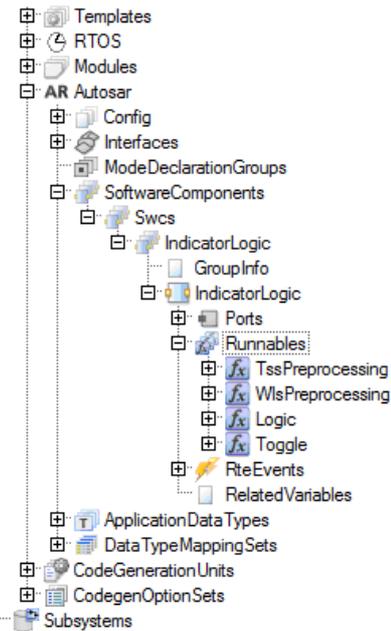


Figure 55: Excerpt from a data dictionary with an imported SWC

3.4 AUTOSAR support in TargetLink

TargetLink offers several functionalities which simplify the behavior modeling according to AUTOSAR. The following sections describes the most important functionalities. You can find a more detailed description in the *TargetLink AUTOSAR Modeling Guide*.

3.4.1 Frame model generation

TargetLink offers a functionality to generate an AUTOSAR frame model, called skeleton mode. This model is based on the AUTOSAR-related specifications of a software component in the TargetLink Data Dictionary (DD). The AUTOSAR frame model generation simplifies designing a Simulink/TargetLink model for a software component significantly, because the TargetLink user does not need to deal with all AUTOSAR details but can focus on filling in the frame model with the actual control algorithm.

To generate an AUTOSAR frame model, select the respective software component object in the TargetLink Data Dictionary.

Choose the **AUTOSAR Tools/Generate Frame Model** context menu entry.

TargetLink then creates a new model from scratch which contains blocks for the runnables, AUTOSAR ports, data access points, etc. for that particular software component.

You can now fill in the content of the runnables with TargetLink blocks, drag already prepared subsystems from a Simulink library into the runnables or e.g. use TargetLink Custom Code blocks to call already existing, non-AUTOSAR legacy C code functions.

In any case, to obtain a first design of the SWC in TargetLink is made rather easy.

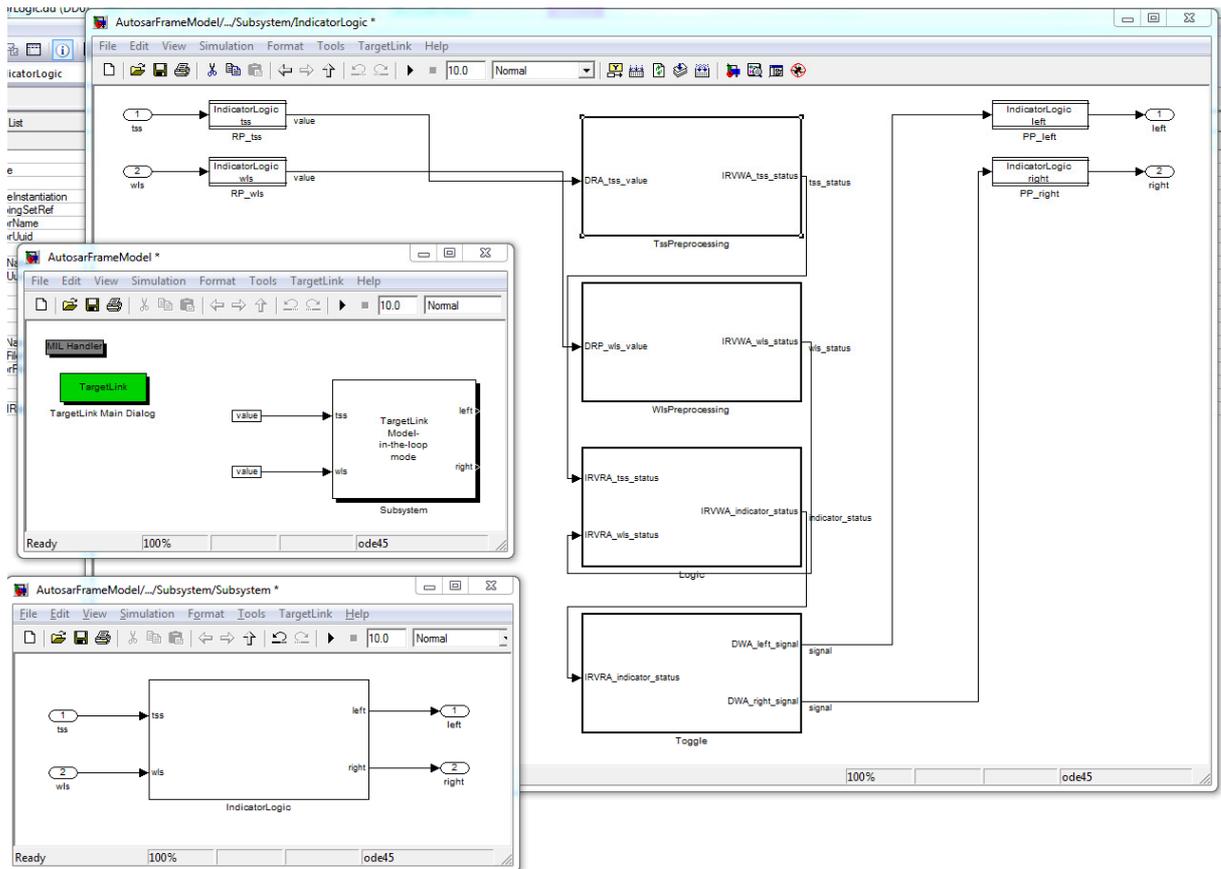


Figure 56: A generated frame model

The frame generation functionality in TargetLink does not only support model generation from scratch, but also the automatic update of an existing model of a SWC.

For that purpose, Select the respective SWC object in the Data Dictionary and choose the **AUTOSAR Tools/Update AUTOSAR Model** context menu entry.

TargetLink then not only updates the model but also generates an update report which contains hyperlinks to navigate to the individual AUTOSAR-related blocks and the respective DD objects. Moreover, the AUTOSAR frame model functionality in TargetLink provides a huge number of hooks to adapt the frame generation to individual company-specific needs.

3.4.2 Generation of application data types or implementation data types

If you use application data types (ADTs) in your software architecture, you can use TargetLink to generate implementation data types (IDTs) and data type mapping sets. These describe internal values and specify implementation details like a base type. IDTs are needed for code and RTE generation and appear in the generated code.

In the context menu of a software component, select **AUTOSAR Tools / ImplementationDataType Creation Wizard**. In the upper part of the dialog, you can select the package in which newly generated IDTs are saved or you can specify the name for a new package.

The lower left part of the dialog shows all ADTs which are not mapped to IDTs in the data type mapping set of the selected software component. If no data type mapping set exists, all ADTs are shown and a new data type mapping set is created.

In the lower right part of the dialog, you can see a suggestion for an IDT for the selected ADT. You can either map the ADT to an existing IDT or base type, or you can generate a new IDT by specifying a new name and a base type. In this case, the scaling and the ranges of the IDT are taken over from the ADT. Click **Apply** to save the settings for the selected ADT.

In a similar way, you can generate ADTs from IDTs using the **Application Data Type Creation** wizard, if this is required.

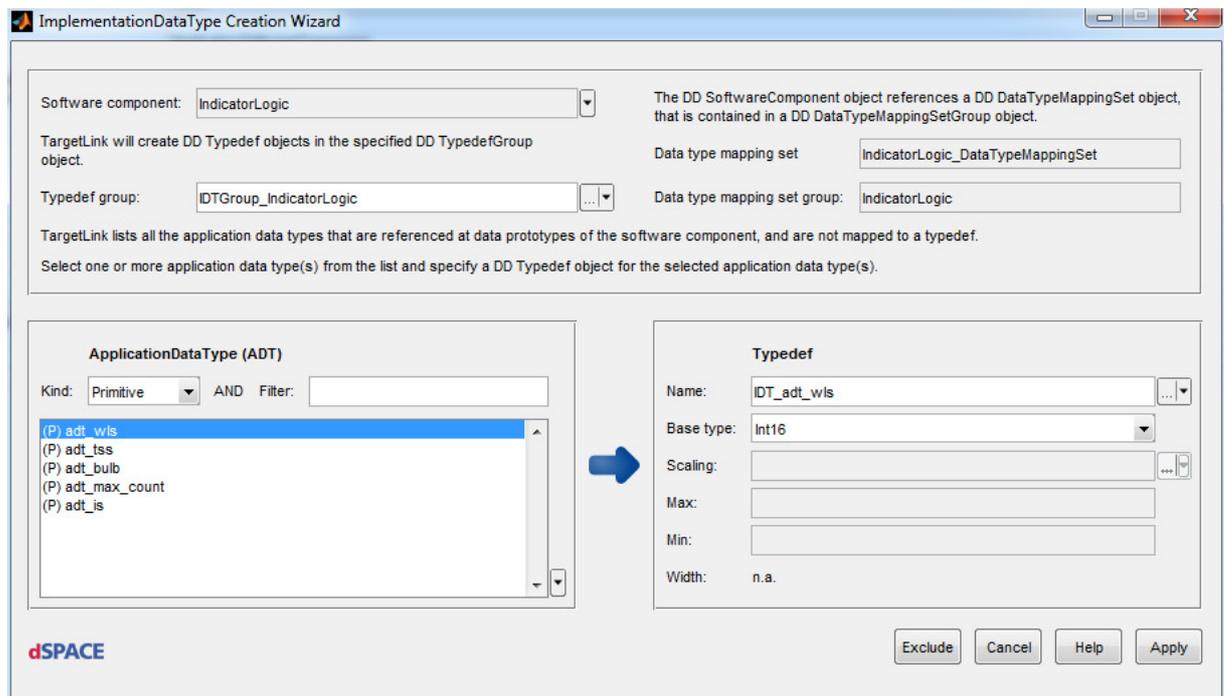


Figure 57: Mapping ADTs to IDTs

Note that each atomic software component needs a reference to a data type mapping set. If you use TargetLink to generate them, you receive an own mapping set for each software component. If you want to use the same mapping set for several software components, create it in SystemDesk and reference it from each software component. You can then generate or map the IDTs in TargetLink using the wizard.

3.4.3 Generating a constant specification mapping

As described in section 2.2.2.3, constants which specify physical values must be transformed into internal values to be used in the code for the software components and for RTE generation. For this, EB tresos Studio expects a mapping of such application constants to implementation constants via a constant specification mapping set.

TargetLink generates such a mapping automatically if the mapping does not already exist. For each application constant, it uses the compumethod (scaling) attached to its data type to compute the internal value, saves it as an implementation constant and maps both constants in a constant specification mapping set. If no compumethod exists, an identical mapping is assumed. Like for data type mappings, TargetLink creates one constant specification mapping set for each software component.

To use the same mapping for several software components:
 Create the mapping set and reference it from the SWCs in SystemDesk.

3.5 Let TargetLink automatically create the implementation constants and the mappings. Delivering of the software component and the code files

When you finished the implementation and code generation, use the container manager to transport the ARXML files and the generated code files to the SystemDesk project. TargetLink creates an AUTOSAR implementation element which references the created code files. Additional AUTOSAR elements may be created in TargetLink, for example:

- Implementation data types and a data type mapping
- A constant specification mapping
- Parts of the internal behavior of a SWC

These AUTOSAR elements are included in the container and imported back to SystemDesk. There they can be used for subsequent steps.

To export a container for one SWC from TargetLink, choose Export Container from the context menu of a software component. You can import it in SystemDesk via the context menu of your project.

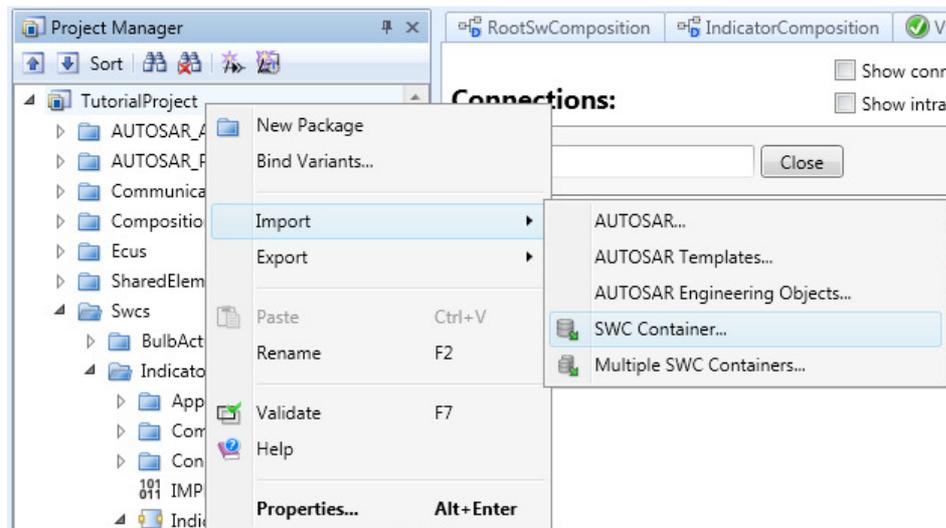


Figure 58: Importing a container in SystemDesk

4 Building the ECU target software

You can now compile and link the basic software generated in section 2.3 and the application software according to section 3.5 using the EB tresos AutoCore build environment. You can find more details in the EB tresos AutoCore documentation. This workflow step can be divided into two steps.

4.1 Preparing the build environment

You must provide the installation path of your EB tresos Studio installation.

Open the file

```
<tresos-installation>/workspace/<your-tresos-project>/util/launch.bat
```

and set the variable `TRESOS_BASE` to the installation path of EB tresos Studio.

You must provide the location of your application files, generated from TargetLink.

Open the file

```
<tresos-installation>/workspace/<your-tresos-project>/util/common.mak
```

and add or extend the path setting of the variable `CC_FILES_TO_BUILD`.

If you are using the EB tresos WinCore, nothing else has to be done.

If you generate the code for an embedded target, you must configure the compiler.

Open the file

```
<tresos-installation>/workspace/<-your-tresos-  
project>/util/<target>_<derivate>_Makefile.mak>
```

and set the variables `TOOLCHAIN` and `COMPILER`.

Then open the file

```
<tresos-installation>/workspace/<your-tresos-  
project>/util/<target>_<derivate>_<compiler>_cfg.mak>
```

and set the variable `TOOLPATH_COMPILER`.

4.2 Building the software

You can now build the complete software, including the basic software and the application. Ensure that EB tresos Studio is closed when you start the build process for the first time.

Then do the following steps:

- Execute
`<tresos-installation>/workspace/<-your-tresos-project>/util/launch.bat`
- Type `make depend` in the DOS shell
- Type `make` in the DOS shell

After compilation, you can find the executable file within the folder:

`<tresos-installation>/workspace/<your-tresos-project>/output/bin`

You can now load the executable file to the target with your debugger tool chain.

5 Runnable mappings exchange

It is common that the application software developer also creates the mapping of runnables to OS tasks. Technically the runnable mapping is part of the ECU configuration, it is contained in the OS and RTE configuration modules. The contents of these modules are vendor-specific and described via vendor-specific module definitions, also called *parameter definitions*.

The application software developer is able to define runnable to task mappings using SystemDesk.

SystemDesk fully supports import, editing and automated creation of runnable to task mappings for vendor-specific module configurations.

After the runnable mappings are created, the resulting OS und RTE module configurations are exported from SystemDesk and merged into the EB tresos Studio project. **Error! Reference source not found.** displays an overview of the process.

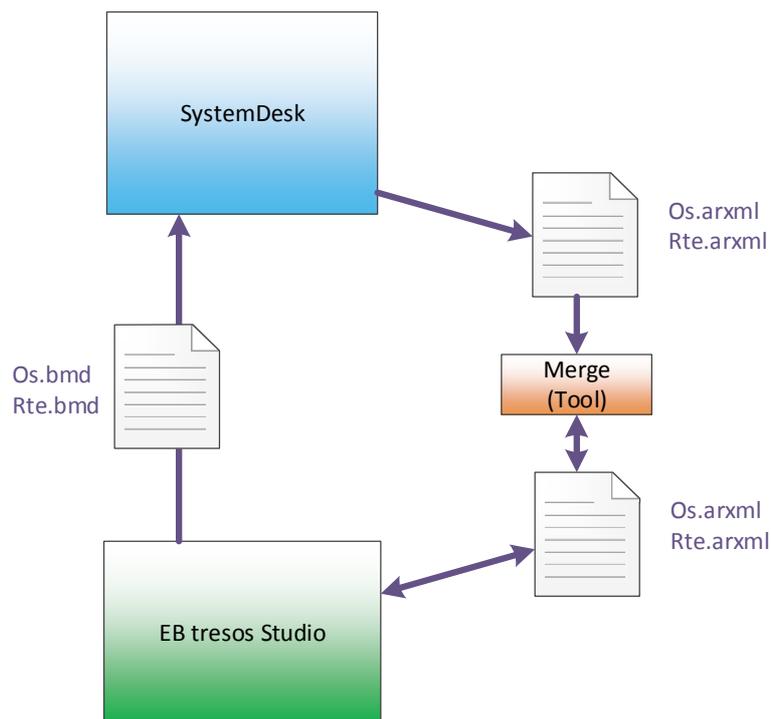


Figure 59: Runnable mapping exchange process

The merging of the runnable mappings on XML level is quite difficult to read and thus error prone. The console-based tool `MappingsMerger.exe` is provided with this application note. It automates the merging of the runnable mapping relevant data into an existing EB tresos project.

Hint:

It is possible to export the module configurations from EB tresos Studio, import them into SystemDesk and after adding the runnable mappings export them back to EB tresos Studio. However, that approach is more complicated to handle because it exposes all module configuration details beyond the runnable mappings to the application software developer. Also the merging back to EB tresos Studio is more complex.

5.1 Creating runnable mappings in SystemDesk

5.1.1 Step 1: Importing EB tresos Studio parameter definitions into SystemDesk

The EB tresos Studio provides its own version specific parameter definitions in files with the BMD extension. You can obtain the parameter definitions for the OS and RTE module configurations from the EB tresos Studio installation folder. The specific locations are:

- <tresos-installation>/ACG<AcgVersion>/plugins/Rte_TS_<Version>/autosar/Rte.bmd
- <tresos-installation>/ACG<AcgVersion>/plugins/Os_TS_<Version>/autosar/Os.bmd

You can locate them easily by searching for Os.bmd and Rte.bmd in the EB tresos Studio installation folder. Multiple search results indicate that the module configuration is available for multiple

AUTOSAR versions.

Select the highest version if no specific AUTOSAR version is prescribed for your project.

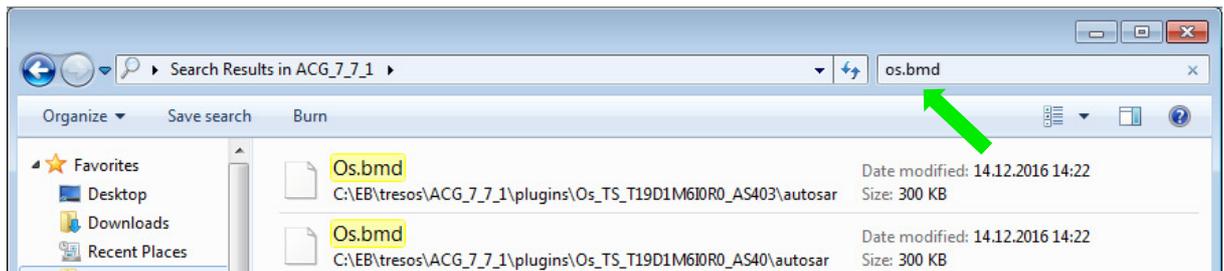


Figure 60: Example on how to locate os.bmd

It is recommended to copy the Os.bmd and Rte.bmd files into the folder structure of the SystemDesk project to locate them more easily.

5.1.2 Step 2: Creating RTE and OS module configurations

Create an ECU configuration in SystemDesk first to create an EB tresos-Studio-specific RTE configuration.

For details, see the SystemDesk Manual -> *Configuring ECUs*.

Select **Create empty ECU** configuration in the New ECU Configuration dialog to create an ECU configuration to configure BSW modules.

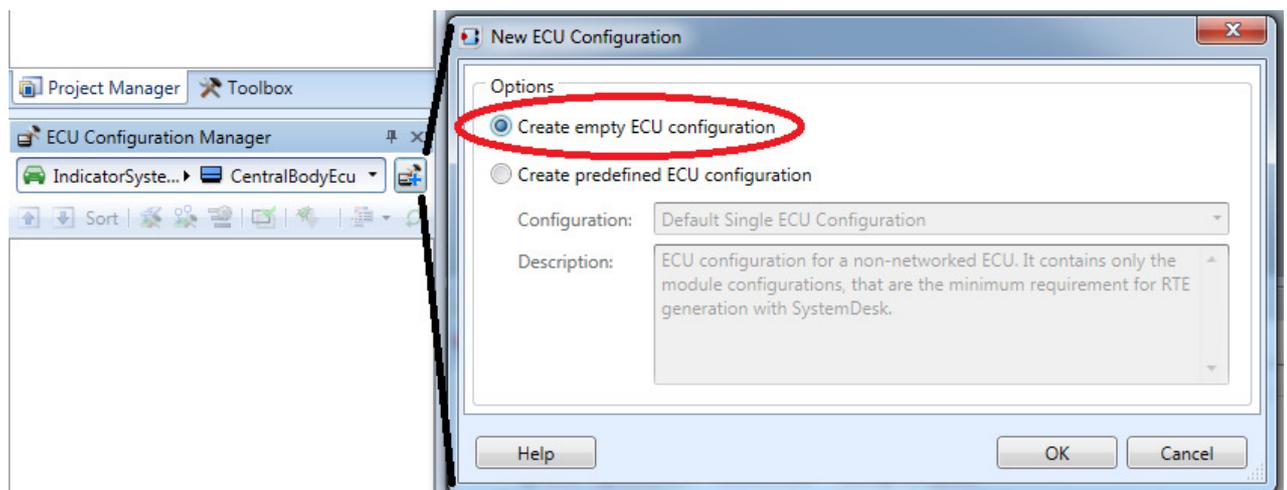


Figure 61: Creating an empty ECU configuration

To add a new module configuration to the ECU Configuration.

Choose **ECU Configuration -> New -> Module Configuration**.

Select the `Rte.bmd` parameter definition file as depicted in Figure 62.

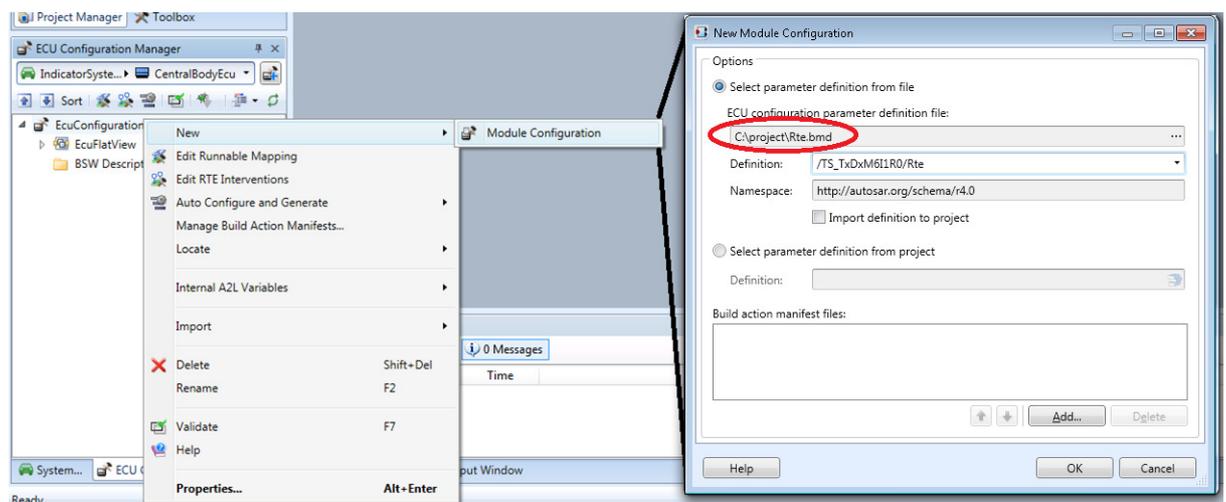


Figure 62: Creation of the RTE module configuration

To create runnable to task mappings, an OS configuration is needed.

Add a new module configuration to the ECU Configuration.

Select the `Os.bmd` parameter definition file in the New Module Configuration dialog, see Figure 63.

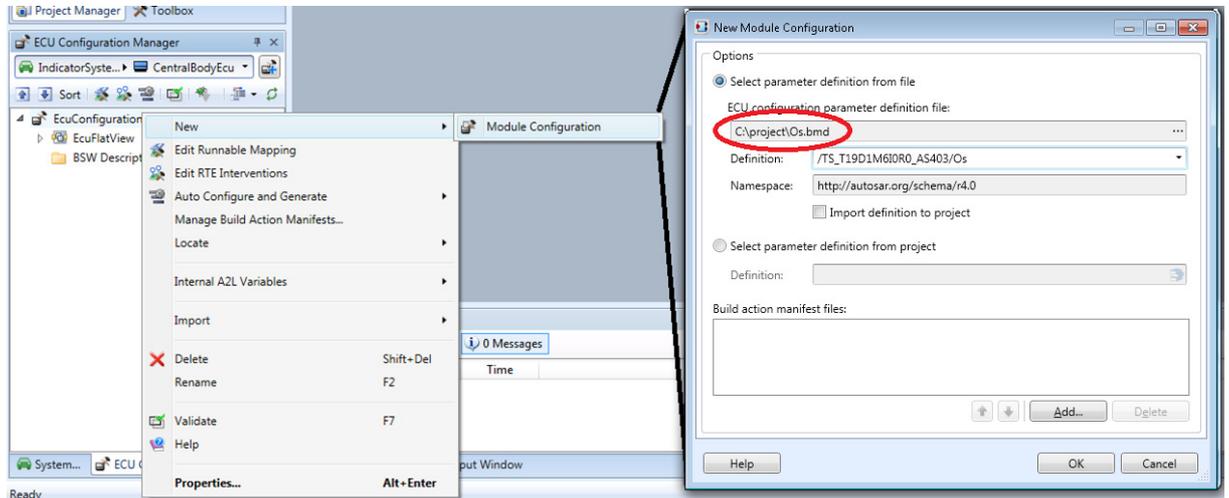


Figure 63: Creation of the OS module configuration

Start the SystemDesk's Runnable Mapping editor via the tool bar button **Edit Runnable Mapping**, see Figure 64.

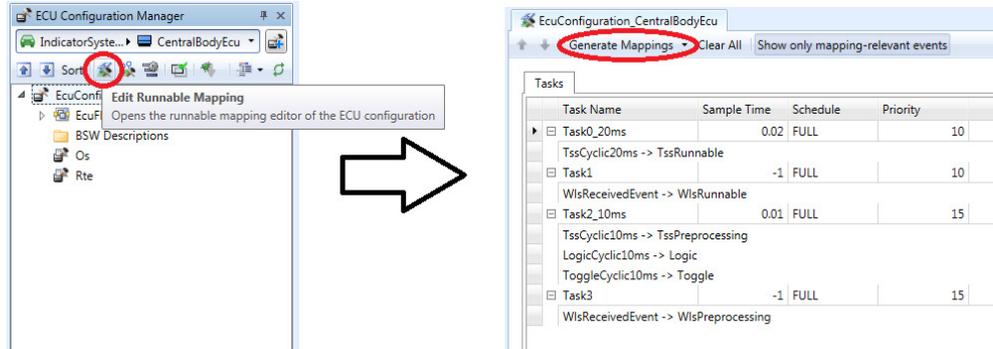


Figure 64: Creation of runnable mappings

Use the automated Generate Mappings feature of the editor to create initial runnable to task mappings and then adjust the tasks and mappings as needed.

5.1.3 Step 3: Unifying the ECU extract top level composition

The ECU Configuration in SystemDesk is based on a so called *ECU Extract of the System Configuration Description* with a flattened top level composition.

Packages of an ECU configuration are not directly visible in SystemDesk by default, but they are shown in **the Project Manager**.

Set the option in the Project Manager.

Choose **Tools->Preferences->ECU Configurations->Show ECU configurations in Project Manager**.

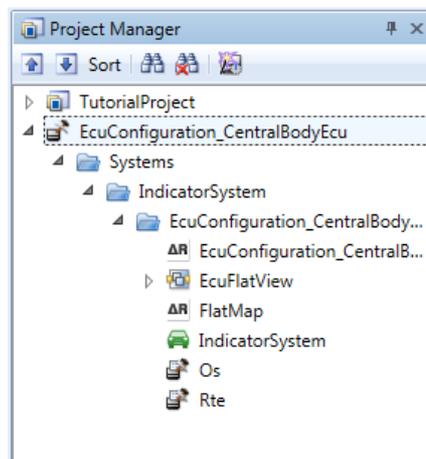


Figure 65: Package structure of the ECU configuration

Within an ECU Configuration created by SystemDesk, software components from the `EcuFlatView` flattened top level composition are referenced, see Figure 66:

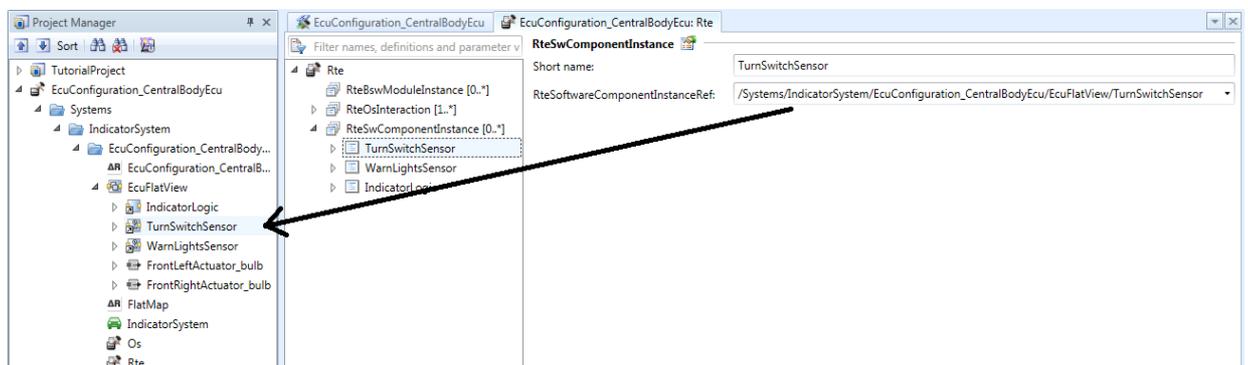


Figure 66: Example of a module configuration reference to the ECU extract

However, this ECU Extract is not exchanged with EB tresos directly. But the System Extract is provided to EB tresos Studio. A new ECU extract is generated in EB tresos Studio which is used as reference point for the ECU configuration.

The package structure of the ECU Extract created in EB tresos Studio differs from the default package structure that is generated in SystemDesk, see Figure 67.

It is necessary to unify the package structure of the ECU Extract in SystemDesk with that created by EB tresos Studio in order to keep the references contained in the RTE module configuration valid.

The ECU Extract package structure and top level composition created by EB tresos Studio, see Figure 67:

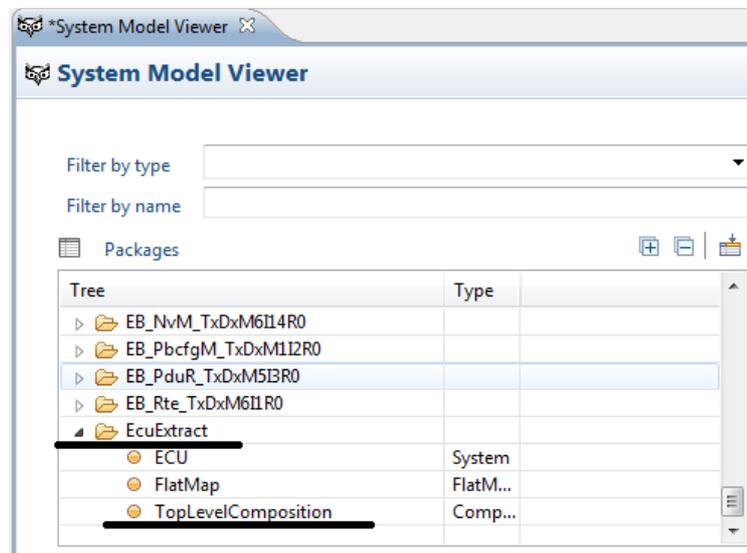


Figure 67: EB tresos Studio ECU extract package structure

To unify the structures, rename the composition EcuFlatView in SystemDesk to TopLevelComposition, see Figure 68:

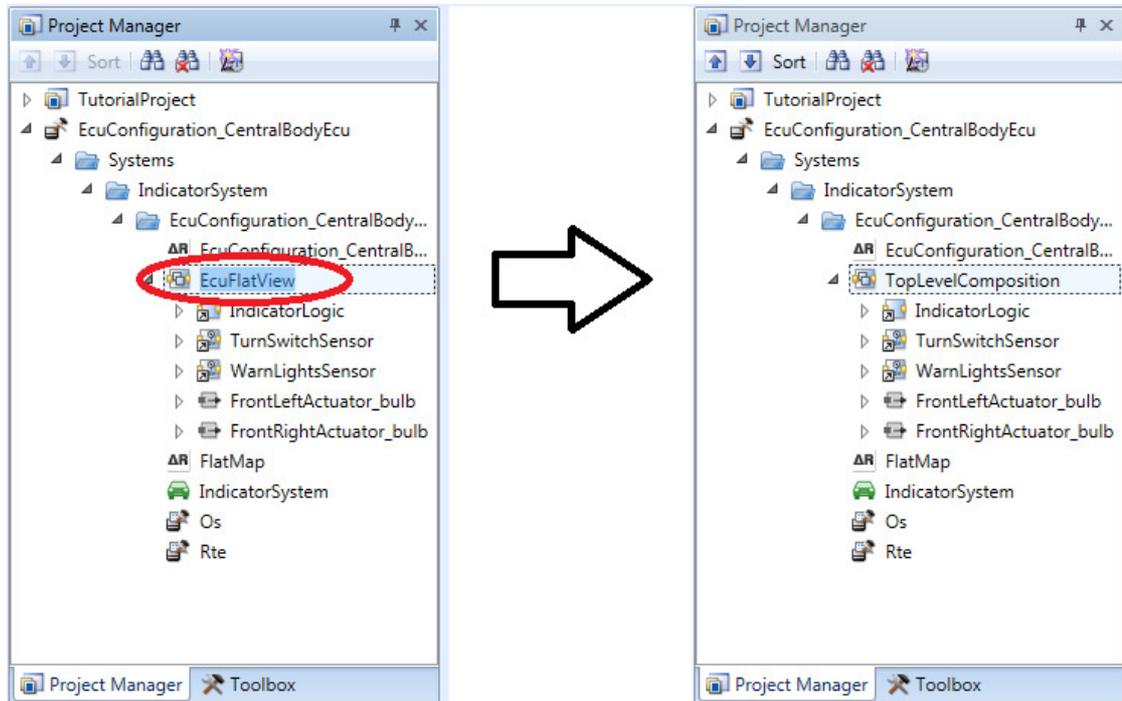


Figure 68: Renaming the top TopLevelComposition

Create a new AUTOSAR package in the ECU configuration node in the Project Manager, see Figure 69:

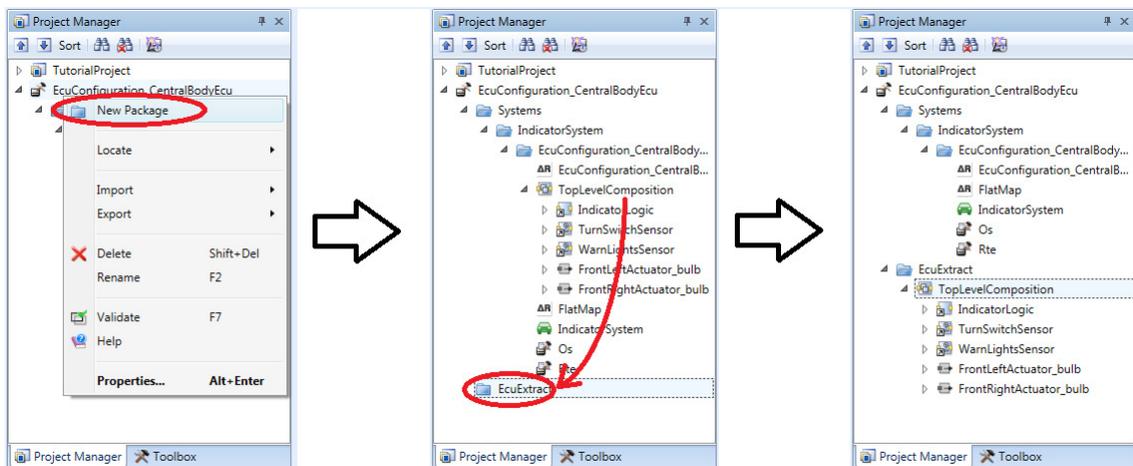


Figure 69: Restructuring the TopLevelComposition

Set the name of the created package to `EcuExtract`.

Use drag and drop to move the `TopLevelComposition` into this package.

The references in the RTE module configuration are automatically adapted to the modified package structure, see Figure 70:

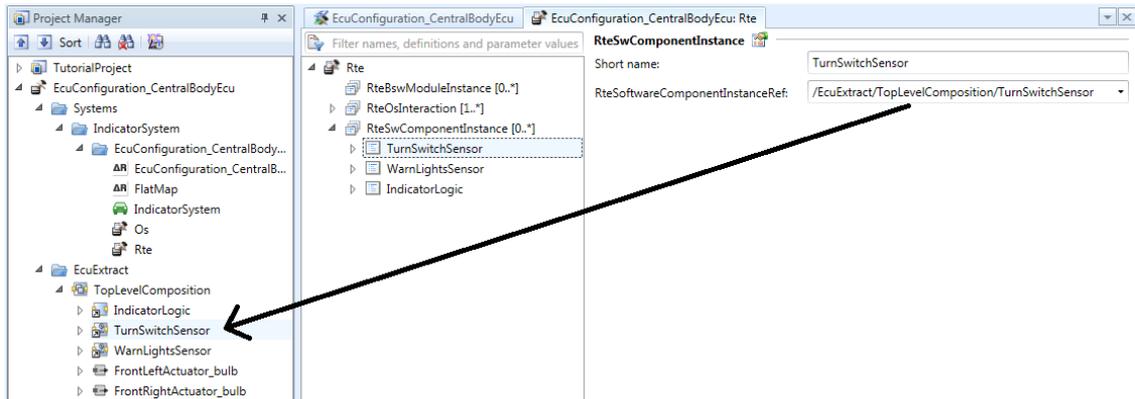


Figure 70: SystemDesk adjusts references properly

You must adjust the location of the Os and Rte module to simplify the subsequent merging process. Create the packages `Os` and `Rte` under the ECU configuration node, see Figure 71:

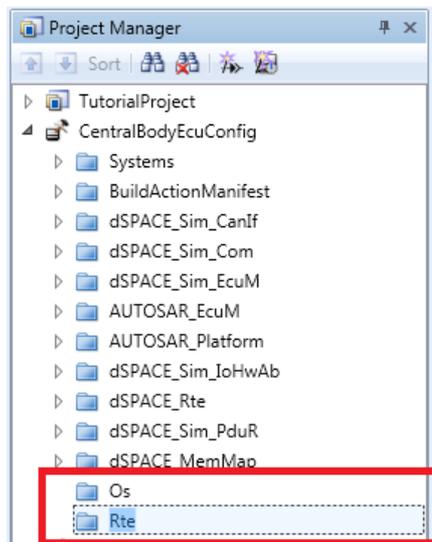


Figure 71: Creating new packages for Os and Rte

Move the module configurations Os and Rte into the packages Os and Rte:

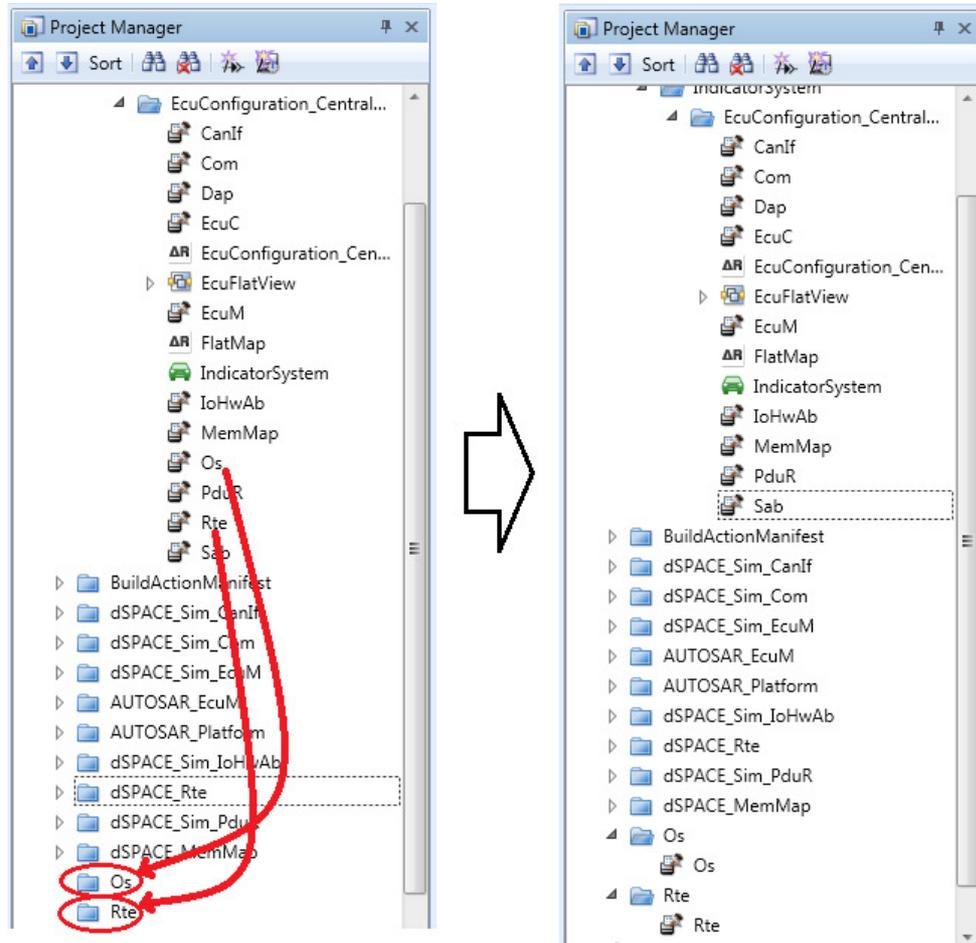


Figure 72: Relocation of the Os and Rte module configurations

The module configurations are now ready to be exported to EB tresos Studio.

5.1.4 Step 4: Exporting the OS and RTE node configurations

You can start the AUTOSAR export of any module configuration from the ECU Configuration Manager by calling the context menu of the module.

Choose **Export -> AUTOSAR**, see Figure 73.

Export the Os and Rte module configurations into two different files, preferably `Os.arxml` and `Rte.arxml`.

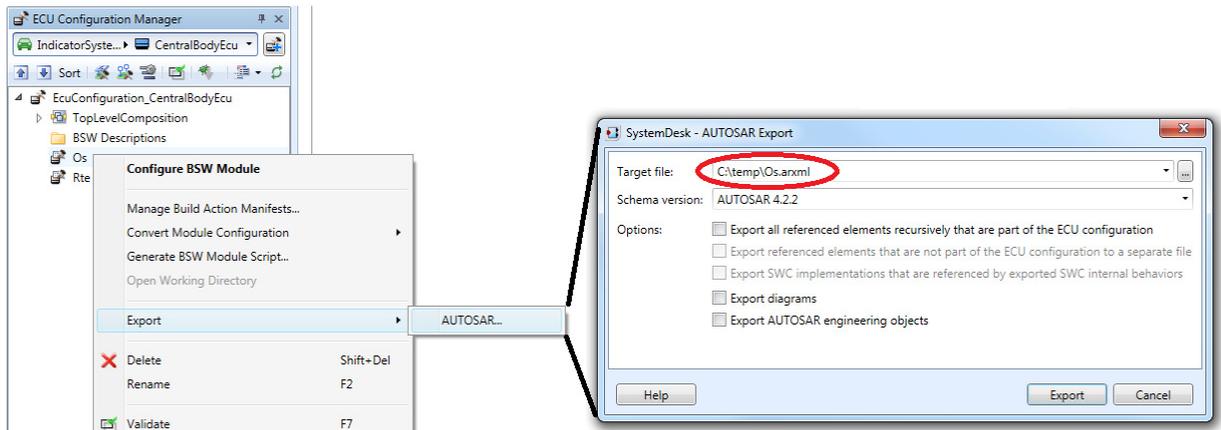


Figure 73: Example how to export the Os module configuration

5.2 Merging the runnable mappings into EB tresos Studio

The next step is to merge the runnable mappings which are contained in the Os and Rte module configuration and which are exported from SystemDesk into the existing project.

To do this, export the current module configurations for Os and Rte from EB tresos Studio into ARXML files.

The MappingMerger tool is provided with this application note. The tool merges all relevant data into the exported ARXML files which you can inspect for changes and import back into EB tresos Studio.

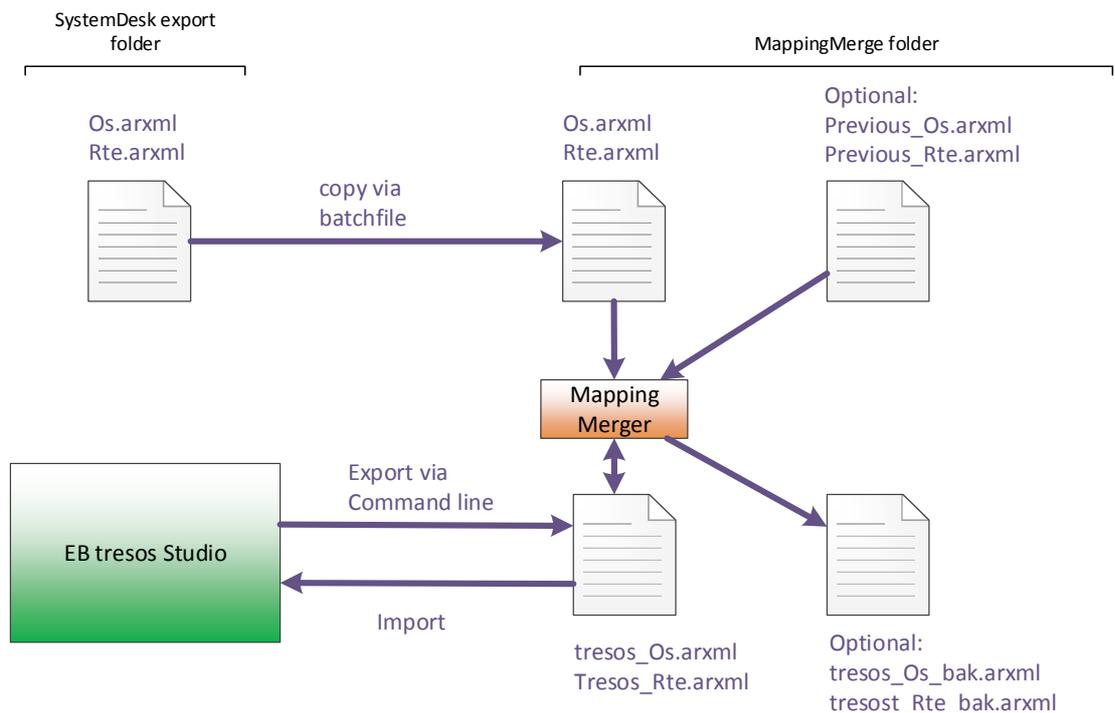


Figure 74: Merging process overview

Hint:

The previous versions of the Os and Rte ARXML files exported from SystemDesk are needed to recognize deleted elements in a newer version of those files. Therefore it is recommended to set up the permanent `MappingMerge` folder as described in section 5.2.1.

The Os and Rte module configurations that are exported from SystemDesk are automatically copied into the `MappingMerge` folder. Any already present Os and Rte module configuration files are re-named and prefixed with `Previous_`. The handling of previous configuration file versions is automated.

By default the original EB tresos Studio files are copied as BAKfiles. This files are useful to inspect differences created by the merging process via XML-diff.

5.2.1 Initial folder setup and batch file adjustment

The merging process is executed via a batch file which is provided with this application note.

You must adjust it to your project folder structure initially then you can use it without further modifications.

It is recommended to create the following folder: <-your-tresos-project>/MappingMerge

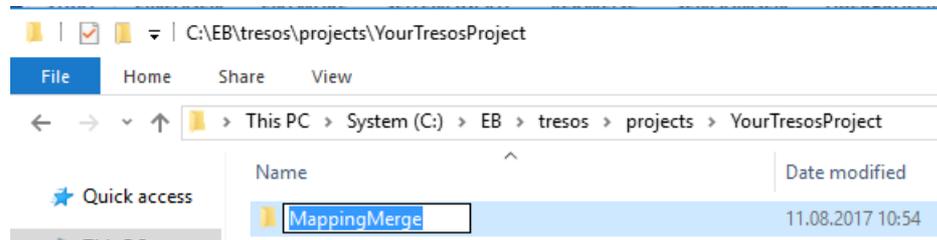


Figure 75: Creation of the mapping merge folder

Copy the files MergeMappings.bat and MappingsMerger.exe into the created folder. The files are provided by this application note.

Hint:

All files needed for the merge process; and all files created by the merge process, are stored in the created folder.

SystemDesk exports all **Os** and **Rte** module configurations to the folder <SystemDeskExport-Folder>.

You must set the path to the created folder in the batch file MergeMappings.bat :

```
1:: dSPACE solution for merging of runnable mappings into an existing tresos project
2:: Copyright 2017, dSPACE GmbH.
3
4:: How to use:
5:: * Create the folder "MappingsMerge" in your tresos project folder
6:: * Copy this batch file into that folder
7:: * Make sure the 'util' folder exists in your project folder and that you have properly set PROJECT_ROOT variable in the launch_cfg.bat
   file
8:: * *** Set the SystemDeskExportFolder variable to the folder where the SystemDesk module configuration files were exported to, see line
   13 ***
9:: * Remember to close tresos before calling this batch file
10
11 @ECHO OFF
12
13 set SystemDeskExportFolder=### Paste the location of the Os.arxml and Rte.arxml exported from SystemDesk ###
14
15:: Adjust these variables if your naming convention differs from that used by here
16 set OsFromSystemDeskName=Os.arxml
17 set RteFromSystemDeskName=Rte.arxml
18 set OsPreviousName=Previous_Os.arxml
19 set RtePreviousName=Previous_Rte.arxml
20 set OsTresosExportName=tresos_Os.arxml
21 set RteTresosExportName=tresos_Rte.arxml
22 set OsTresosBakName=tresos_Os_bak.arxml
23 set RteTresosBakName=tresos_Rte_bak.arxml
24
25:: Set this on to 'OFF' if you do not wish to create backupfiles of the original tresos arxml-files
26 set CreateBackupFiles=ON
```

Figure 76: One time batch file adjustment

Hint:

All file names seen here are defined in the MergeMappings.bat batch file.

You can adjust the names as needed.

Prerequisite on your project configuration: The batch file relies on a properly configured EB tresos Studio project folder structure.

Make sure the following file exists: <-your-tresos-project>/util/launch_cfg.bat

In the file **launch_cfg.bat** you must set the variable `TRESOS_BASE` to the proper EB tresos Studio installation folder – see section 4.1.

5.2.2 Initial creation of the EB tresos Studio importers

So called *importers* are used in EB tresos Studio for module configuration import.

Importers are created in EB tresos Studio via the context menu on the ECU:

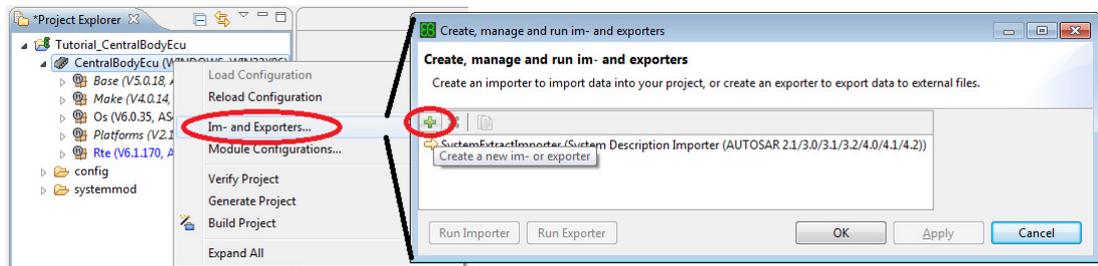


Figure 77: Creation of the RTE module configuration importer

Define a name for the importer, for example `RteImporter` :

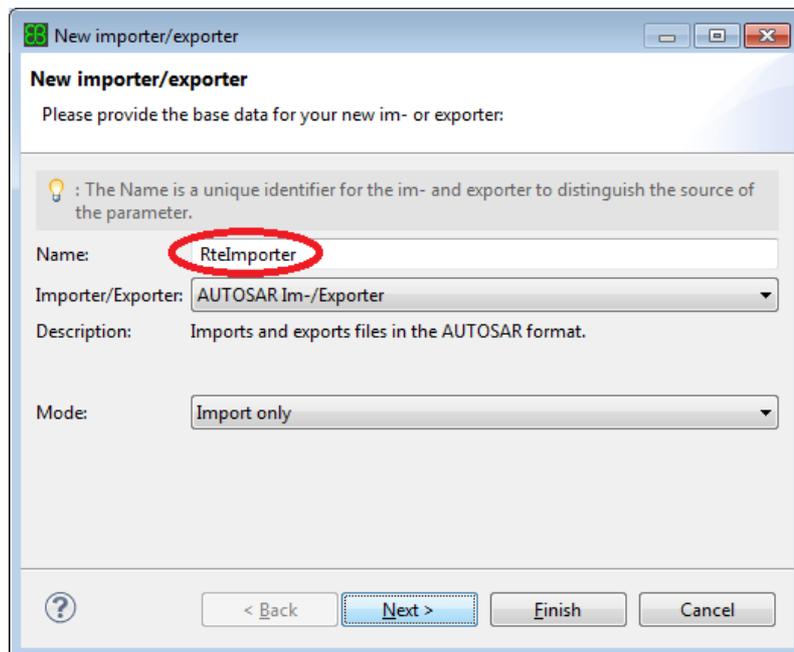


Figure 78: RTE importer overview dialog

The further settings are to be left at the default settings, see Figure 78.

On the next wizard page you must specify the file name and the import Strategy.

Set the **Content type** to `asc:4.2.2`.

Click the checkbox *Validate against AUTOSAR schema*.

Provide in File name the path to the `tresos_Rte.arxml` file in `<-your-tresos-project>/MappingMerge`.

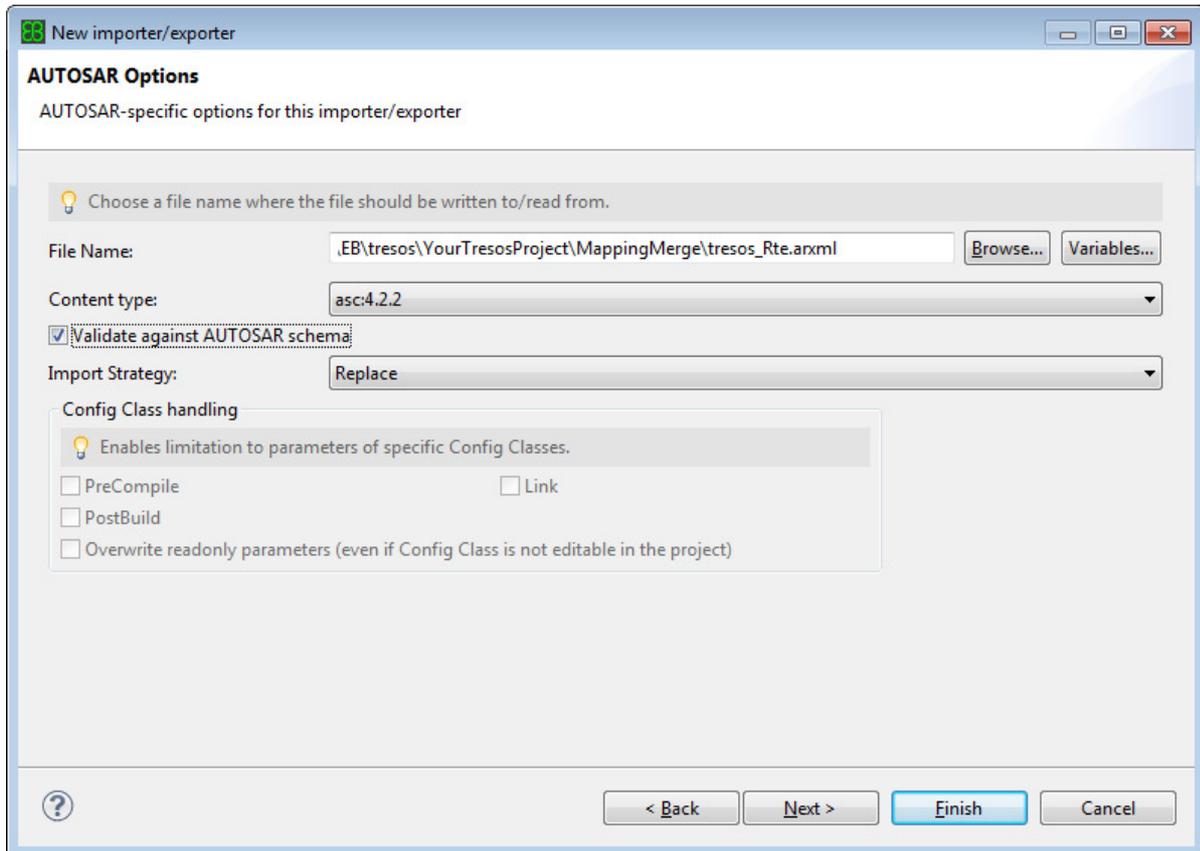


Figure 79: Configuration of the RTE module importer

All further settings may remain at their default values and you can close the wizard with **Finish**.

Repeat this import procedure for the Os module configuration. Once both importers are created and configured, you can run them without further adjustments.

5.2.3 Executing the MergeMappings.bat and Run Importer

EB tresos Studio must be closed before you execute the batch file.

If new Os and Rte files are available in the `<SystemDeskExportFolder>` the merge process starts by executing the `MergeMappings.bat` file. See process description in section 5.2.1.

Check the output of the merge process for error messages in the command line, see Figure 80.

```

Administrator: C:\Windows\system32\cmd.exe
INFO 17-08-14,09:44:36 (2102) Reading file "C:\EB\tresos\YourTresosProject\config\OS.xdm"
INFO 17-08-14,09:44:37 (2101) Writing file "C:\EB\tresos\YourTresosProject\MappingMerge\.\tresos_Rte.arxml"
WARNING 17-08-14,09:44:37 (3143) The preference "Use AUTOSAR-compliant maximum SHORT-NAME length" is deactivated and therefore may produce an invalid AUTOSAR file
Errors "0" Warnings "1"
=> OK

Creating backup files tresos_Os_bak.arxml and tresos_Rte_bak.arxml...
    1 file(s) copied.
    1 file(s) copied.
=> OK

Merging of Os.arxml with and to tresos_Os.arxml
=> OK

Merging of Rte.arxml with and to tresos_Rte.arxml
=> OK

SUCCESS
C:\EB\tresos\YourTresosProject\MappingMerge>

```

Figure 80: Successful run of the MergeMappings batch file

The successful merge result is stored in the files prefixed by `tresos_`.

Use a diff-tool of your choice to compare the merged files with the originally exported files from EB tresos Studio which have the `_bak` postfix in the file name.

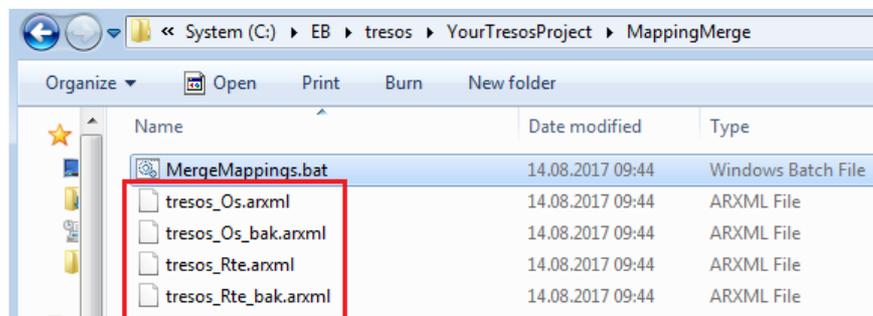


Figure 81: Merged and backup files

After file comparison execute both importers for Os and Rte via **Run Importer**.

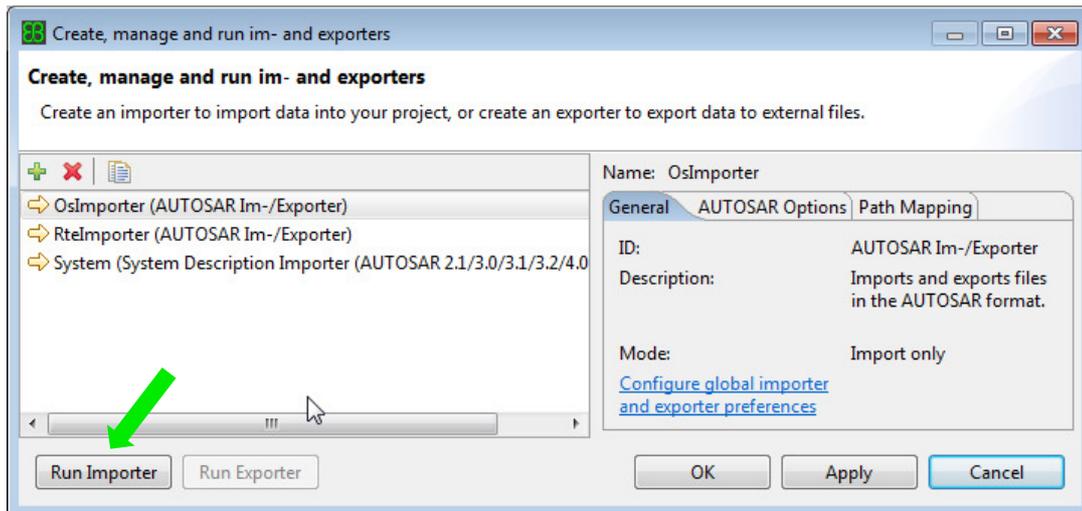


Figure 82: Running the importer

The updated runnable mappings are now available in EB tresos Studio.

6 Detailed element descriptions in SystemDesk

This chapter describes how to model specific elements in SystemDesk, which were described in section **Error! Reference source not found.**

Section 6.1 describes attributes of frequently used elements which are mandatory according to the strict schema.

Therefore model them in SystemDesk, since most of them are needed in EB tresos Studio later on. Section **Error! Reference source not found.** describes how to do a data type mapping in SystemDesk.

Section 0 describes how to model a constant specification mapping.

6.1 Mandatory attributes

This section gives detailed information about attributes of some frequently used elements in SystemDesk which are mandatory according to the strict schema and which therefore are present when you import a project in EB tresos Studio. You can use it as a reference when you design your architecture in SystemDesk.

This document gives information about mandatory elements, but it does not explain them. You can find a description and an explanation of the element in the AUTOSAR templates. A short explanation is also available as a tooltip in the advanced tab of each dialog.

Hint:

SystemDesk presets all mandatory properties with a meaningful default value if possible.

Note that a checkbox has three possible values in SystemDesk. If such an attribute is mandatory, it must be true or false:

<input type="checkbox"/>	Not defined
<input checked="" type="checkbox"/>	True
<input type="checkbox"/>	False

This table lists the mandatory attributes for frequently used elements.

Element	Mandatory attributes
Application array data type	Element type ref Maximum number of elements
Application record data type	At least one record element
SWC internal behavior	At least one runnable entity
Sender/receiver interface	At least one variable data prototype
Client/server interface	At least one operation
Mode declaration group	At least one mode Initial mode ref
Mode switch interface	Mode group
Variable data prototype / parameter data prototype	Data type
Argument data prototype	Data type
Provided/required port prototype	Interface
NonQueued sender ComSpecs	Init value
NonQueued receiver ComSpecs	Alive timeout
Queued receiver ComSpecs	Queue length
Mode switch sender ComSpecs	Queue length
Server ComSpecs	Queue length
Constant specification (application constant)	UnitRef (Advanced page ValueSpec / SwValueCont / UnitRef)
ECU	At least one CommController. Depending on the

type of the CommController, further attributes are required. SystemDesk's strict schema validation can be used to find out which attributes.

System

System version

6.2 Data type mapping

In order to model the data type mapping in SystemDesk, create a new data type mapping set via the context menu of a package: **New / New SWC-T / Data Type Mapping Set**.

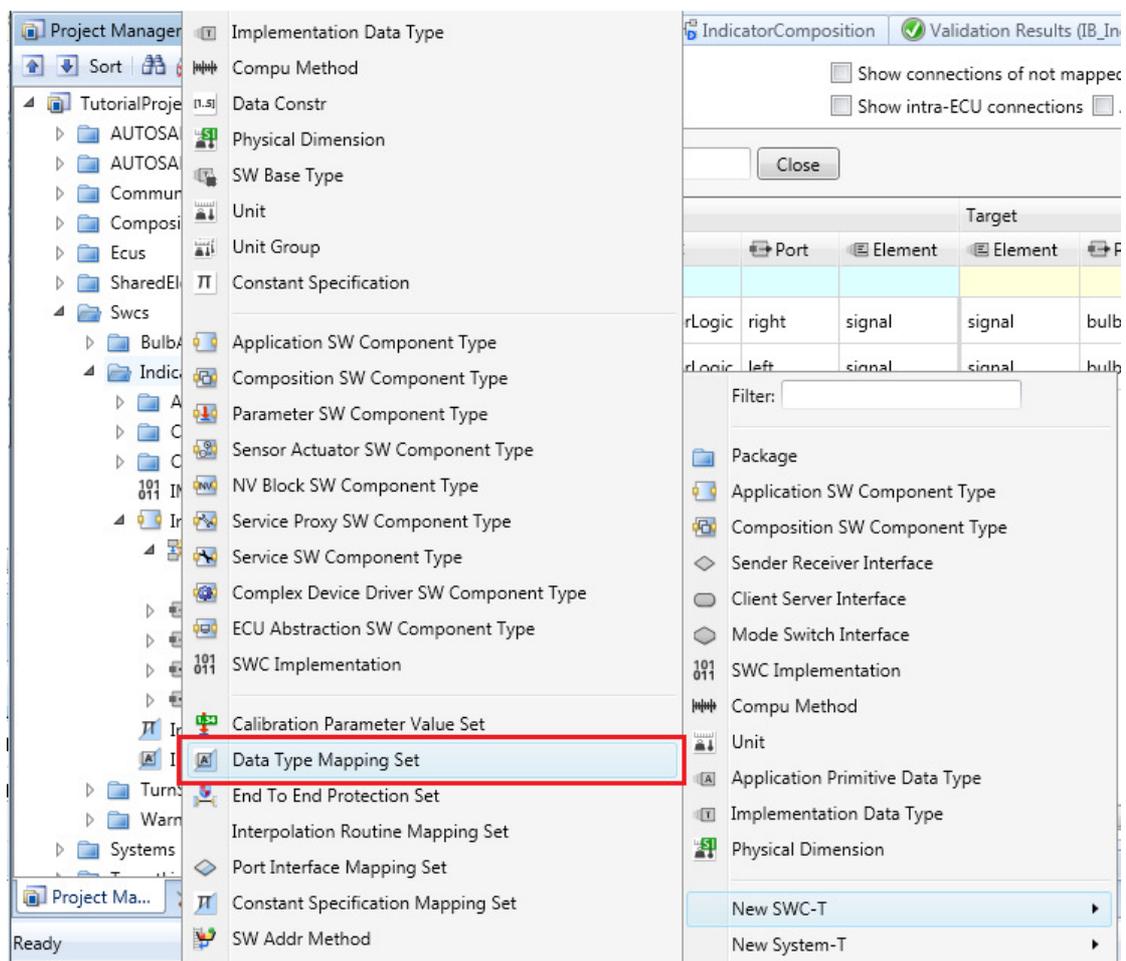


Figure 83: Creating a new data type mapping set

In its **Properties** dialog, you can create new mappings on the **Data Type Mapping** page. Each mapping refers to one ADT (application data types) and one IDT (implementation data type), which must be compatible, i.e. the category and the size.

Once you created one or more data type mapping sets, create a reference from each internal behavior to the mapping(s) which shall be used in this context.

You can do this on the Data Type Mapping Refs page of the internal behavior's **Properties** dialog. You can reference the same mapping set from several internal behaviors, or you can create an own mapping set for each internal behavior. See Figure 84.

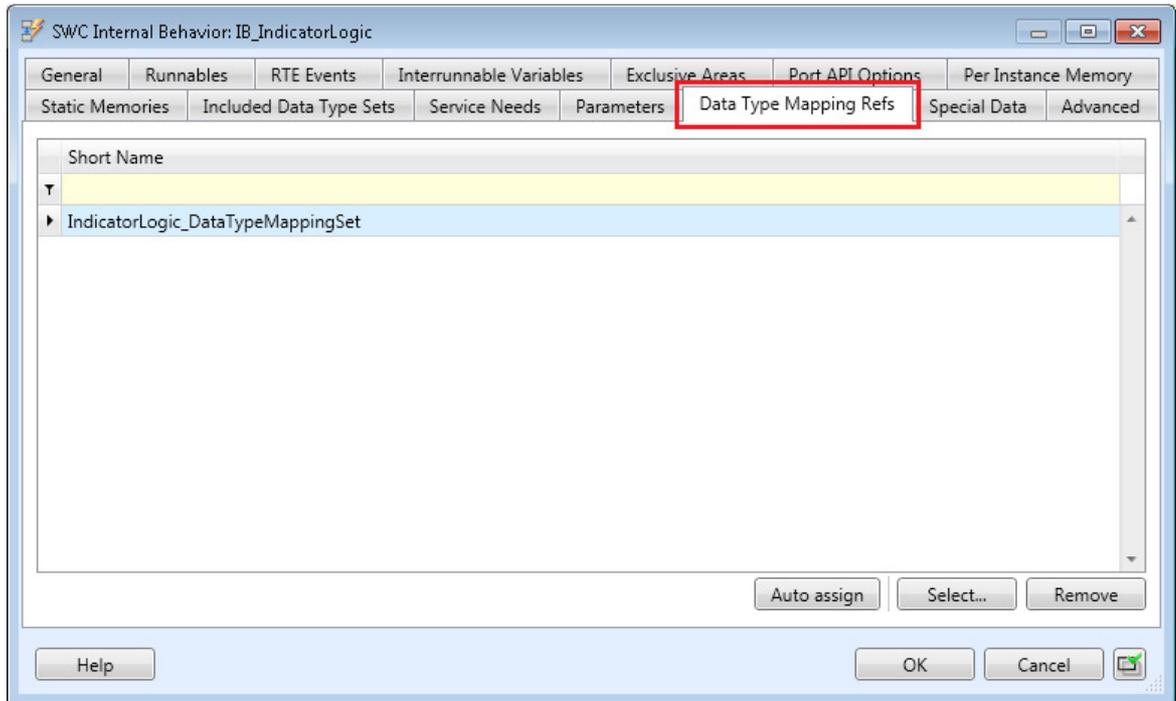


Figure 84: Referencing a data type mapping set from an internal behavior

If you start with an empty data type mapping, it is possible to add all ADTs used by a software component to the mapping set automatically.

Select the data type mapping set to be populated and click **Auto assign**.

Now you have data type mapping set filled with all ADTs used by a specific software component.

Now you must assign an IDT to every application data type in the data type mapping set. SystemDesk provides automatic assignment of the *smallest* IDT to the selected ADTs.

Select all ADTs in the data type mapping set dialog and click **Auto assign**.

An IDT selection dialog pops up.

Check the desired IDTs that shall be used by the automated assignment algorithm as displayed in Figure 85. After you confirmed the selection dialog the best fitting IDTs are assigned automatically.

Hint:

New array and structure IDTs are created as needed.

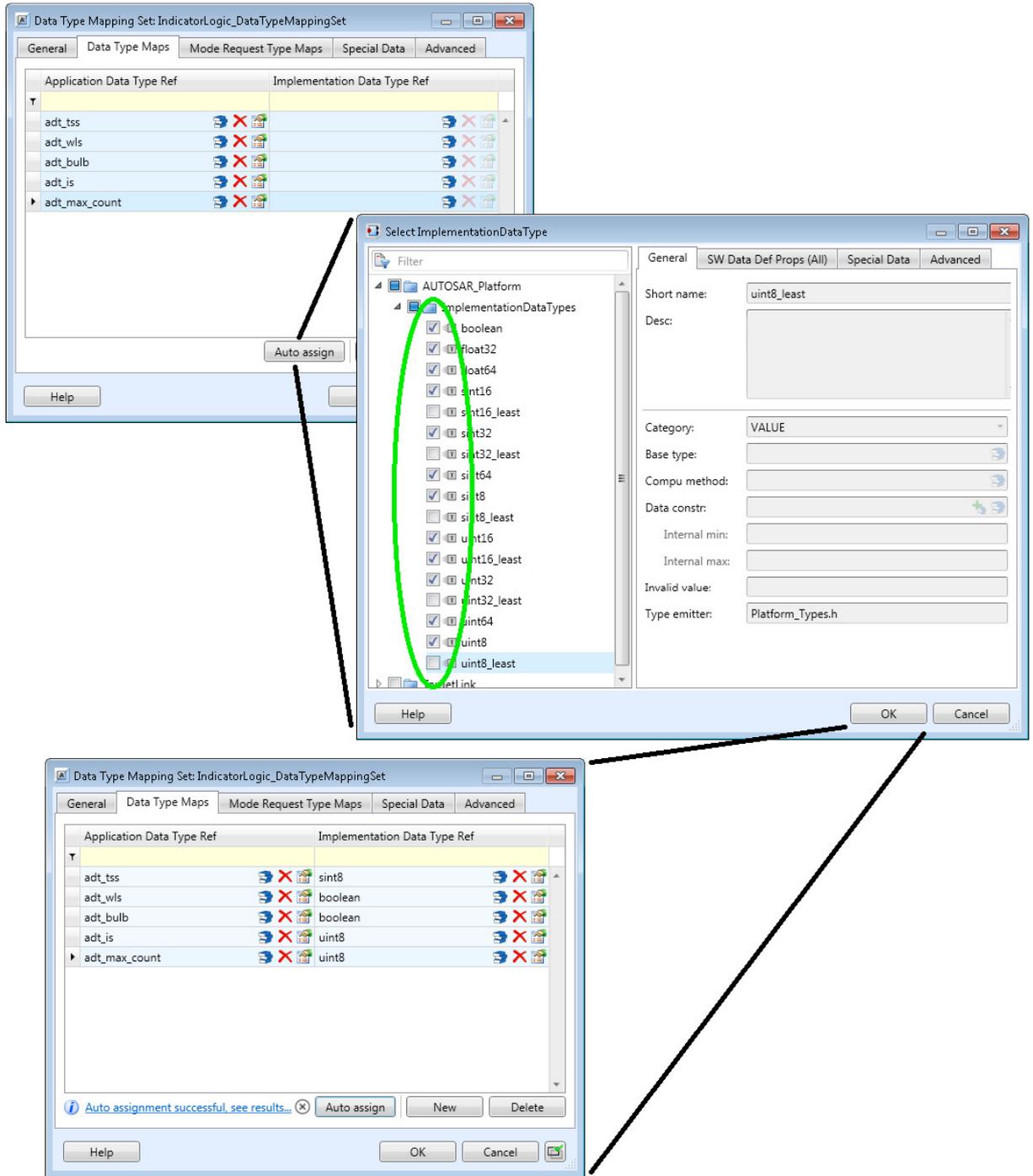


Figure 85: Automatic assignment of IDTs to ADTs

6.3 Constant specification mapping

Before giving a detailed description about how to map application constants to implementation constants, the following figure explains what needs to be done. It is assumed that you mapped your application data types to implementation data types. Now, for each application constant – that is, a constant created for a data prototype which is typed by an application data type – an implementation constant must be created.

As the two constants shall be mapped later on, they must be compatible. SystemDesk assists you in achieving this by letting you choose an implementation data type for the constant, and then SystemDesk generates the internal structure of the constant according to the data type. The easiest way is to choose the implementation data type to which the application data type is mapped.

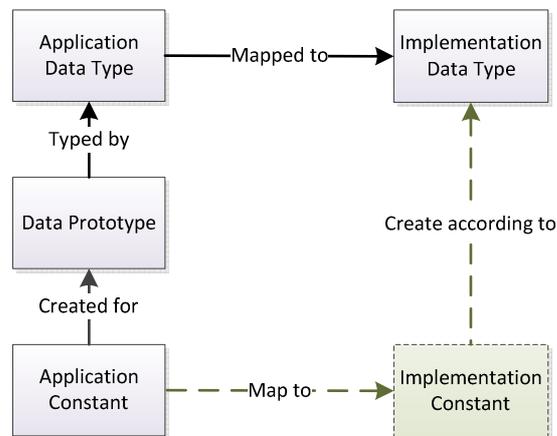


Figure 86: Overview over Constant Specification Mapping

For mapping application constants to implementation constants in SystemDesk, take the following steps. For each application constant, create a new constant specification, and open its **Properties** dialog. Then click the **+** button in the field **Value specification**.

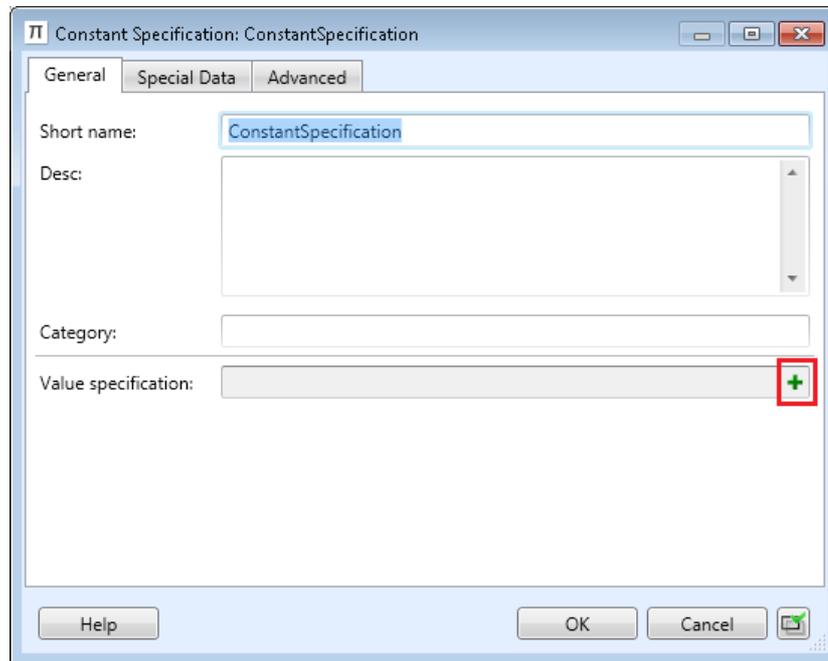


Figure 87: Empty Constant Specification

SystemDesk now opens a dialog.

Select a data type according to which the internal structure of the constant is created. This is where you define whether the constant specification is an application or an implementation constant.

Select the implementation data type to which the application data type of the application constant is mapped.

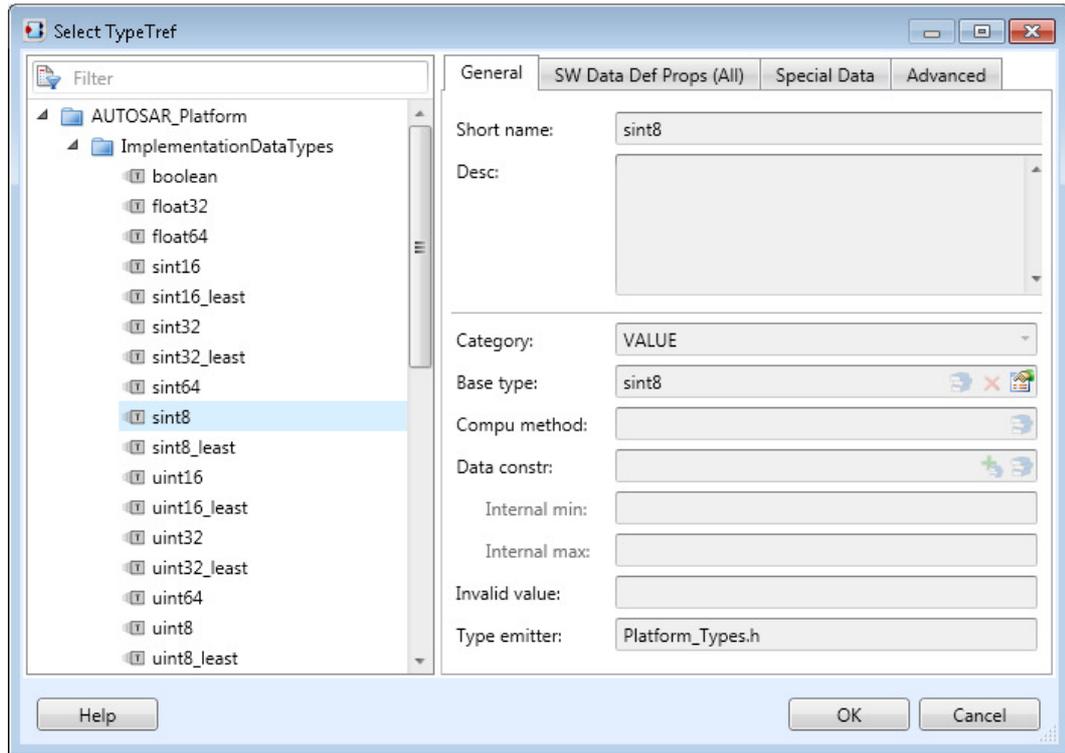


Figure 88: Selecting a data type for a constant

Click **OK**, and a second dialog opens.

Here, you can enter the value of the constant in the field **Scalar**. Note that the attribute `ShortLabel` does not change the short name of the constant which is visible in the **Project Manager**, but the label of the value specification, which can only be seen on the **Advanced** page.

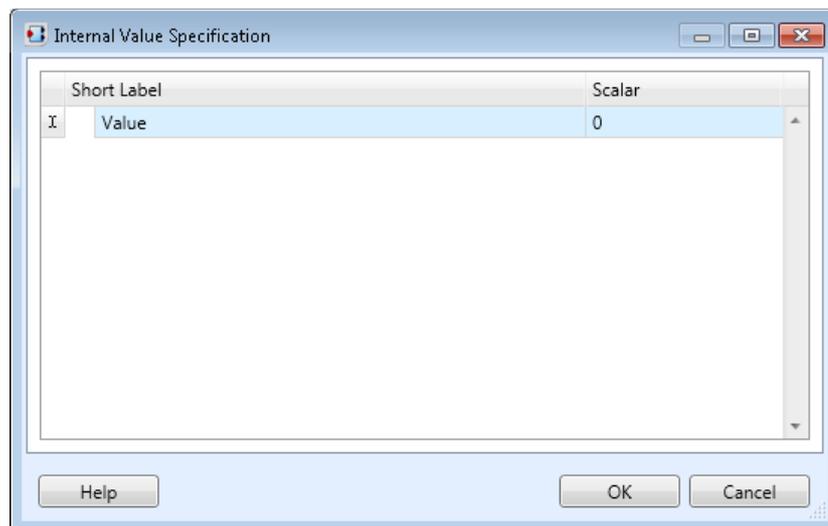


Figure 89: Entering the value of a constant

The implementation constant is complete.

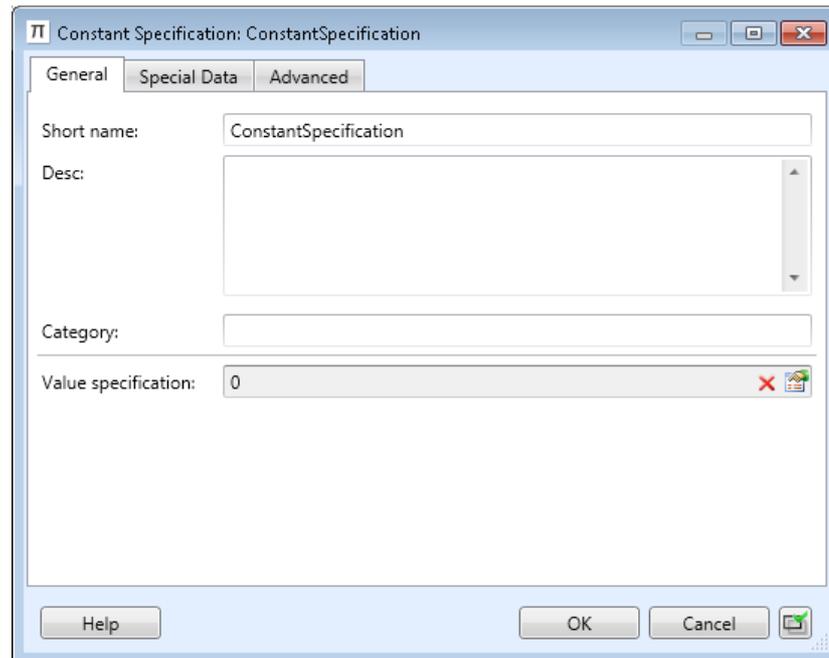


Figure 90: Implementation constant with value 0

When you created implementation constants for all your application constants, the next step is to map the constants to each other.

You must create constant specification mapping sets which contain the mapping of the constants.

You must reference the mapping sets from the internal behaviors of the software components.

AUTOSAR is rather flexible here: Each internal behavior can reference one or more mapping sets, and each mapping set may be used in one or more internal behaviors.

The only constraint is that for each internal behavior, each application constant may only be mapped once to an implementation constant, such that the mapping is unambiguous. This means that you can decide whether you want to create only one mapping set and reference it from all software components, whether you want to create one specific mapping set for each software component, or whether you even want to split the mapping sets.

It is for example possible to create one mapping set for all constants which are shared between several SWCs, and an additional mapping set for each SWC which contain the constants only used in this context.

You can create a new mapping set via **New / New SWC-T / Constant Specification Mapping Set** in the context menu of a package.

In its **Properties** dialog, go to the **Advanced** page.

In the *Mappings*-list, create a new `ConstantSpecificationMapping` by click  to create a new `ConstantSpecificationMapping`. There you can reference the application and the implementation constant.

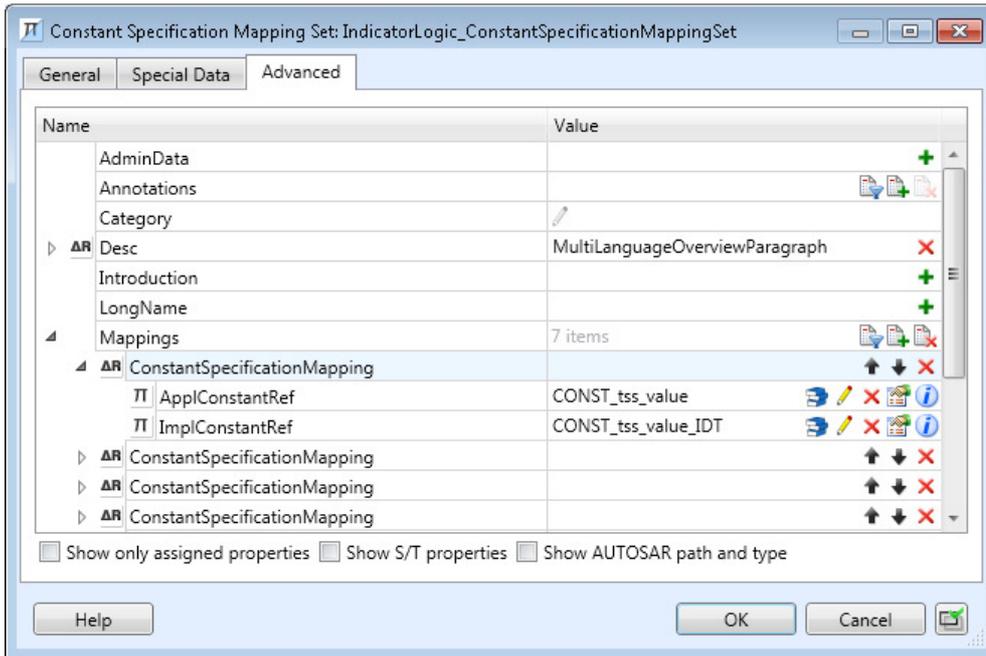


Figure 91: Mapping application to implementation constants

To reference the mapping set from an internal behavior, open the internal behavior's **Properties** dialog and go to the advanced page again. The mapping sets are referenced from `ConstantValueMappingRefs`. To add a new reference, click .

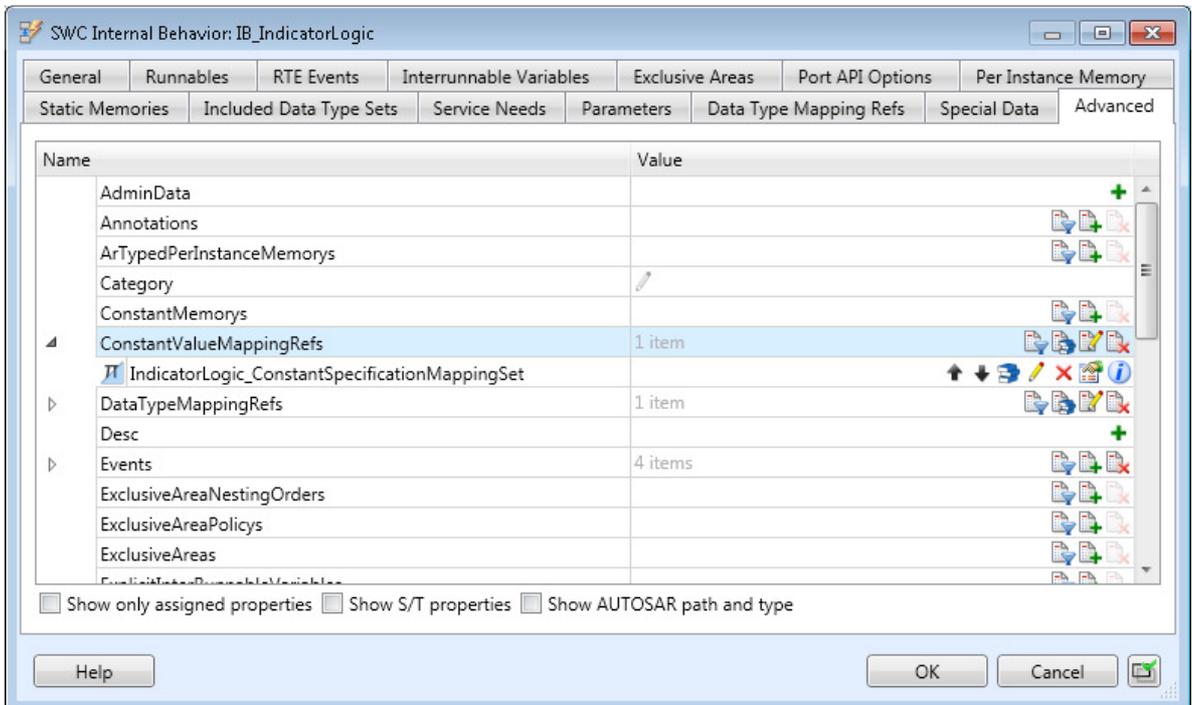


Figure 92: Referencing constant specification mappings from an internal behavior