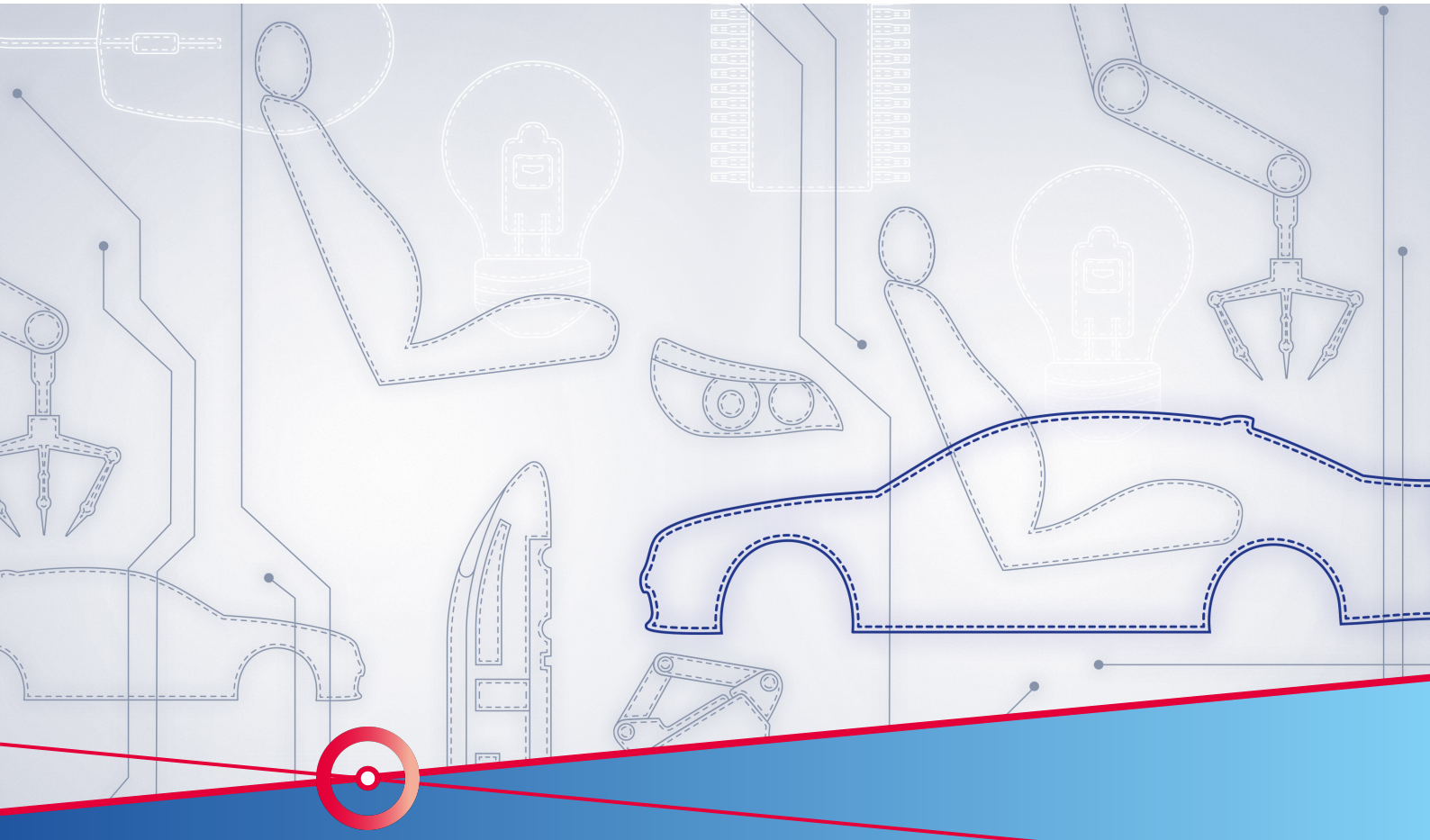
ETAS

DRIVING EMBEDDED EXCELLENCE



RTA Application Note

Add NvM configuration to Project

RTA-CAR

Copyright

Document: AN-000/EN-09-2019

Contents

1 Introduction

1.1 Scope

This application note describes how to integrate into an existing ISOLAR project the Memory Stack configuration using PIMs. The document contains an explanation step by step of the workflow to follow to obtain a working project able to be executed on a virtual target. A sample project is provided along with this document configured as shown in this application note. The starting project used to create this AN is the one obtained following the AN “Project from scratch”.

1.2 Definitions and Abbreviations

BSW: AUTOSAR Basic Software, Hardware independent service layer

RTE: AUTOSAR Real Time Environment

OS: AUTOSAR Operating System

SWC: Software Component

1.3 Conventions

The following typographical conventions are used in this document:

‘Choose **File –> Open**’ –> Menu commands are shown in boldface

Click **Ok** –> Buttons are shown in boldface

Press –> Keyboard commands are shown in angled brackets

The “Open File” dialog box is displayed –> Names of program windows, dialog boxes, fields, etc. are shown in quotation marks.

Select the file `setup.exe` –> Text in drop-down lists on the screen, program code, as weel as path- and file names are shown in the Courier font.

1.4 Assumptions

You must have the following ETAS software installed:

- ISOLAR-AB 6.0.1
- RTA-BSW 5.0
- RTA-OS 5.6.4
- RTA-RTE 6.8
- ISOLAR EVE 3.2.2

This application note could be used as a reference for any generic AR based project.

2 Tool overview

2.1 ISOLAR-AB

ISOLAR-A is the AUTOSAR Authoring Tool that assists users in designing application software to AUTOSAR standards. It provides a graphical interface to generate, import and modify arxml describing the SWC and System design. ISOLAR-A provides also feature to automatically create system configuration importing legacy files (such as DBC and LDF) and support arxml merger. ISOLAR-A output is the set of arxml containing the System Design configuration.

ISOLAR-B is BSW Configuration tool based on a single domain model (ARTOP) architecture, providing a number of advantages when building AUTOSAR system and ECU software. ISOLAR-B output is the set of EcuC Values containing the BSW modules configuration.

2.2 RTA-RTE

RTA-RTE generates the RTE code (*.c and* .h files). The RTE generator takes in input the SWC and System description generated via ISOLAR-A and provides in output the RTE implementation that will be integrated in the software project.

RTA-RTE outputs include:

- OS configuration required to sustain the System configuration (`OsNeeds.arxml`) to be imported in RTA-OS to maintain automatically aligned System and OS configuration (the OS configuration shall be completed in RTA-OS)
- Measurement and calibration description (McSupportData) that can be imported in a A2L generator supporting AUTOSAR.

2.3 RTA-BSW

RTA-BSW provides the following main features:

- **RTA-BSW ConfGen** uses the System Description and ASW Configuration(s) to create a default BSW Configuration using ECUC Value Collection ARXML. RTA-BSW can automatically generate the configuration for the communication stack (Can and Lin supported) and the Memory stack. The generated EcuC values consist in a complete configuration of the interested module (although it's possible to expand the configuration in a later stage) that allows the user to move to the next step, the code generation.
- **RTA-BSW CodeGen** creates the BSW implementation – this includes:
  - Static source code (*.c and* .h) for the configured modules
  - Dynamic source code for the configured module (configuration dependent)
  - BSW module description, AUTOSAR description of the BSW modules in ARXML format (*_BSWMD.arxml and* `_SWCD.arxml`)*
  - Integration code (*.c and* .h), template files to help the user in the prototyping early stages (no production intent); the user shall modify and adapt integration code as per project requirements.

During RTA-BSW ConfGen phase, RTA-BSW generates also part of the MCAL configuration (ARMXL format) needed to sustain the BSW configuration. This configuration can be imported in the MCAL configuration tool to automatically synchronize the two AUTOSAR layers.

2.4 RTA-OS

RTA-OS provides a graphical interface to configure the AUTOSAR Operating System, and generate the relative source code (or linkable library). The synchronization between OS configuration and System configuration (ISOALR-AB) is implemented sharing a subset of

AUTOSAR description file across the tools. The Operating System has dependency to the Hardware target and the C compiler; ensure your RTA installation is compatible with your embedded set-up.

3 Sample project overview

The final project will be a System composed by 2 ECUs (EcuA and EcuB) where only EcuA will be configured. Besides the 2 starting SWCs a new SWC and the memory stack will be added.

Below a short list of the steps described in Chapter 4:

- configuration in ISOLAR-A of a new application SWC: ports, internal behaviour, runnables, events, data access points, PIMs.
- update of the composition
- ECU extract generation
- in ISOLAR-B creation of an EcuC Value Collection
- confgen using RTA-BSW
- udpate of Rte and Os modules with mapping of ASW SWCs on tasks
- codeGen using RTA-BSW
- mapping of BSW modules on tasks
- ECU Extract regeneration
- RTE regeneration

4 Workflow description

4.1 Part 1 - ASW Configuration

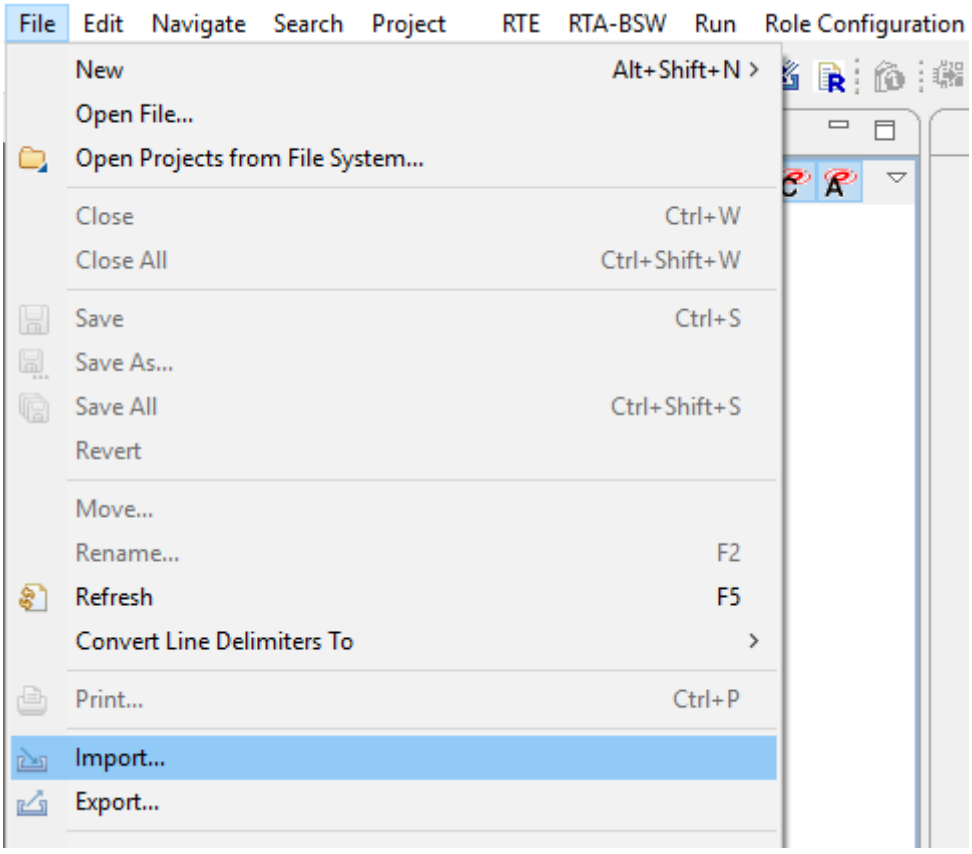
In this first part ISOLAR A is used to configure the ASW layer of the System; a new SWC prototype will be created; it will be executed each 500ms and will take care of reading and writing Non Volatile Memory based on commands received on ports.

NB: ISOLAR A can be accessed by selecting the “AR Explorer” in the Project Explorer view.

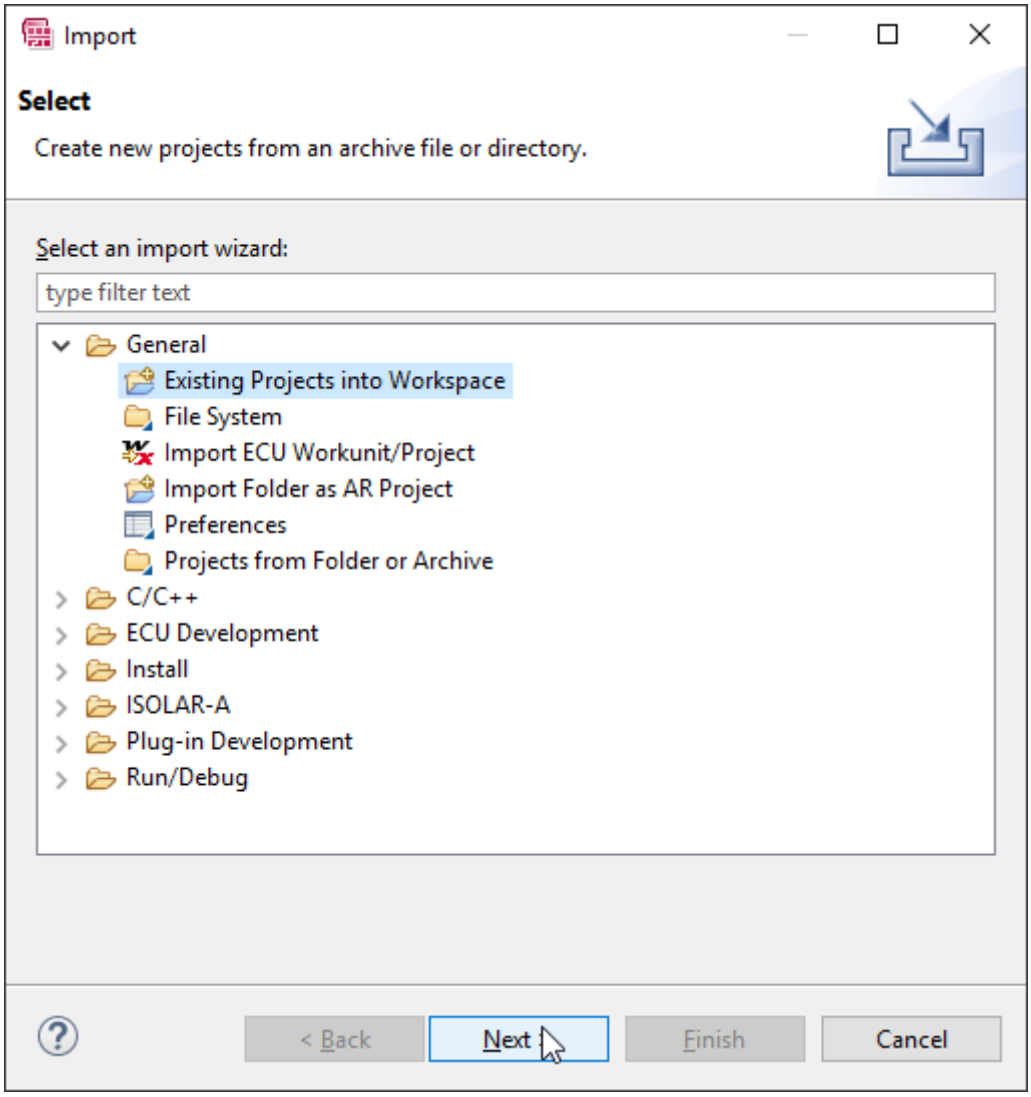
4.1.1 Import an existing AUTOSAR Project

Open a new Workspace in ISOLAR-AB 6.0.1.

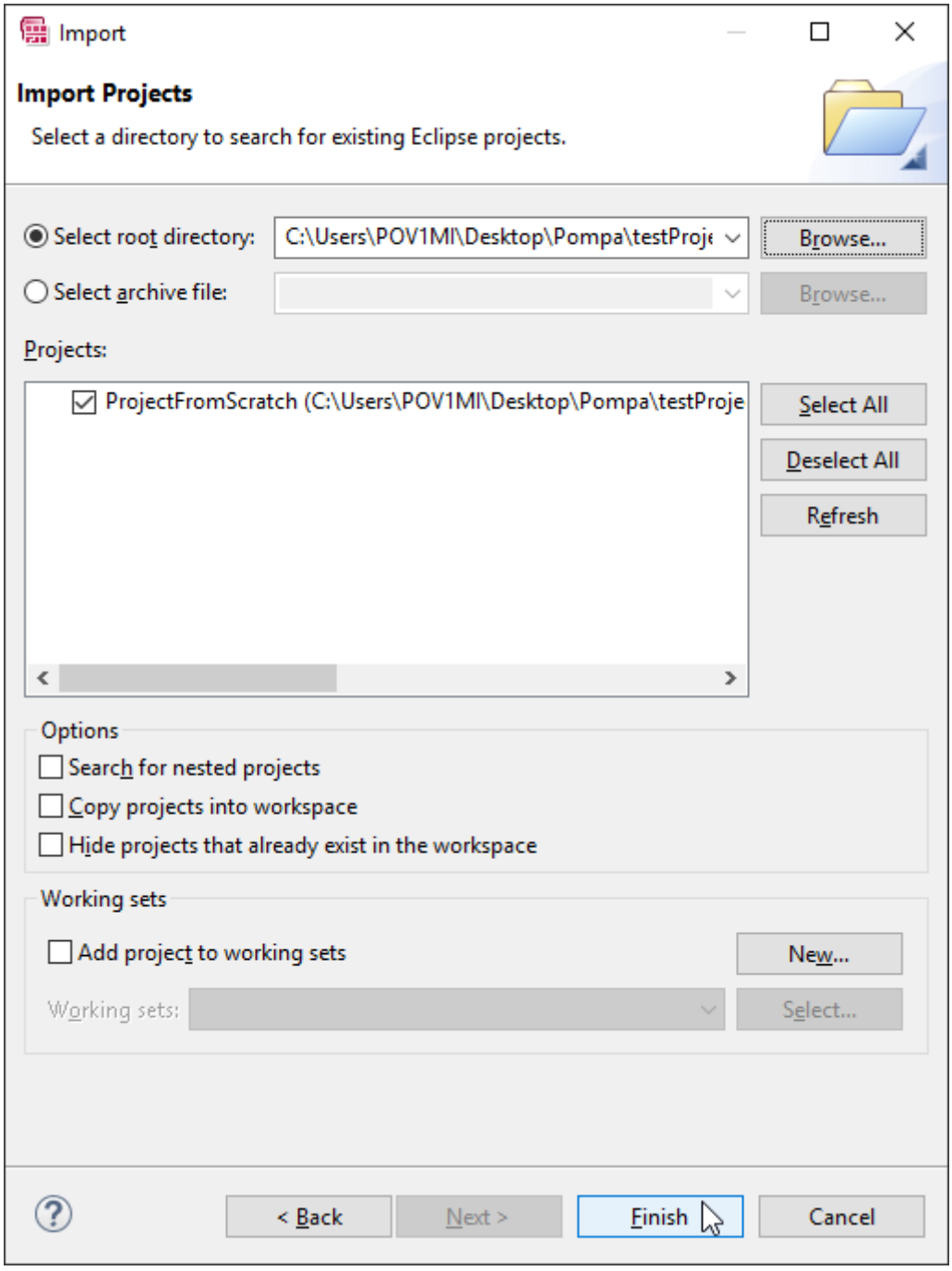
The first step to execute is to import an existing project, in this case the result of the AN “Project from scratch” will be used. To import a project from the Menu Bar select **File –> Import. . .** as shown in Figure



In the pop-up window “Import” choose “General” and then “Existing Projects into Workspace”, then press **Next**.

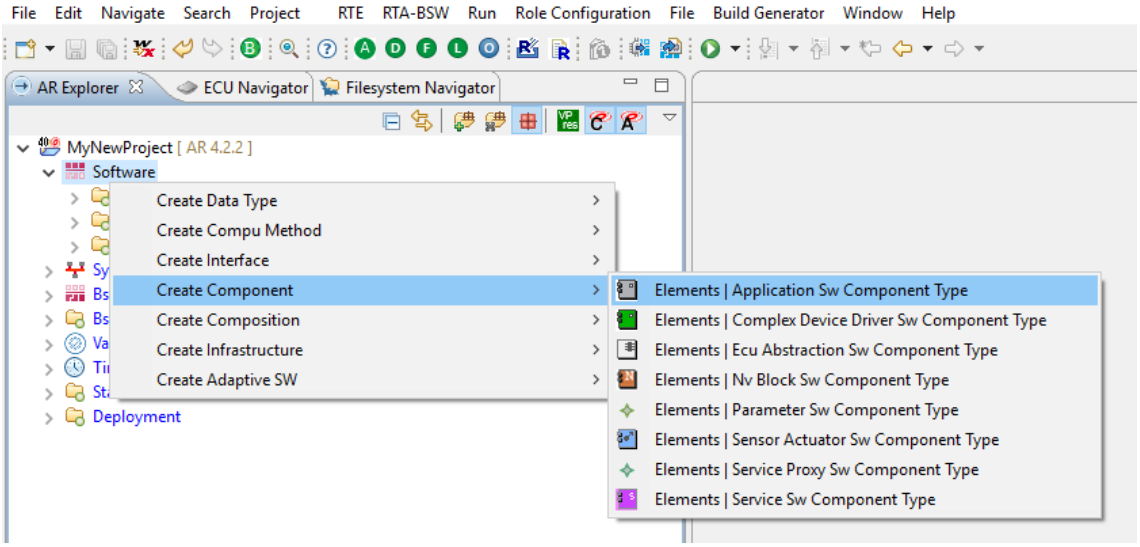


A new pop-up window will appear, choose the archive file or the root directory to the existing project to be imported. If valid project files are found, they are displayed in the box below; select the desired project and press **Finish**.



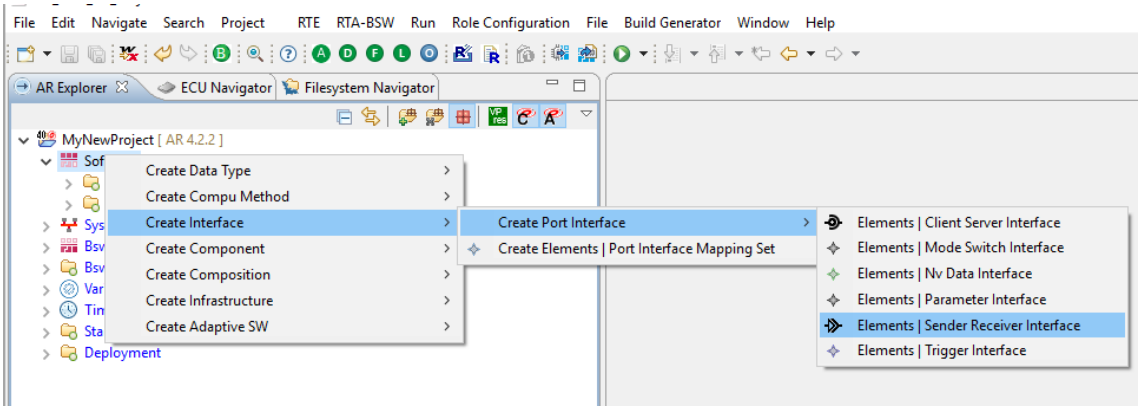
4.1.2 Create a new SWC

Now you can create a new SWC that will be used for NvM handling. To create the SWC right click on “Software” and select **Software –> Create Component –> Elements | Application Sw Component Type** and place the SWC in the ARPackage named “SWCs”.



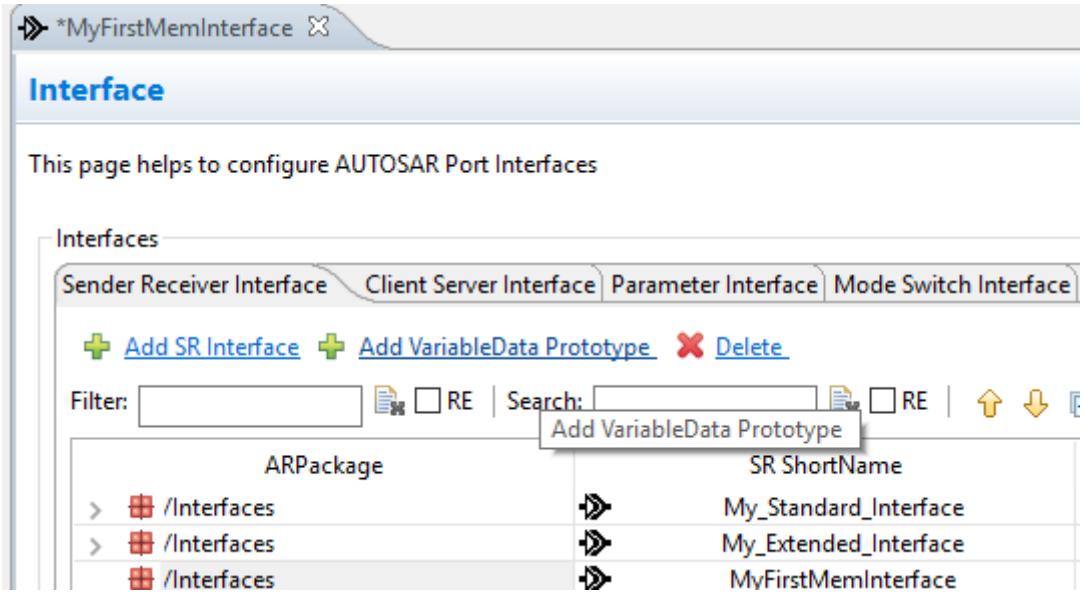
4.1.3 Create a new Port Interface

Ports connecting SWCs must have a reference to a defined Port Interface. To create a new Port interface right click on “Software” and select **Software –> Create Interface –> Create Port Interface –> Elements | Sender Receiver Interface.

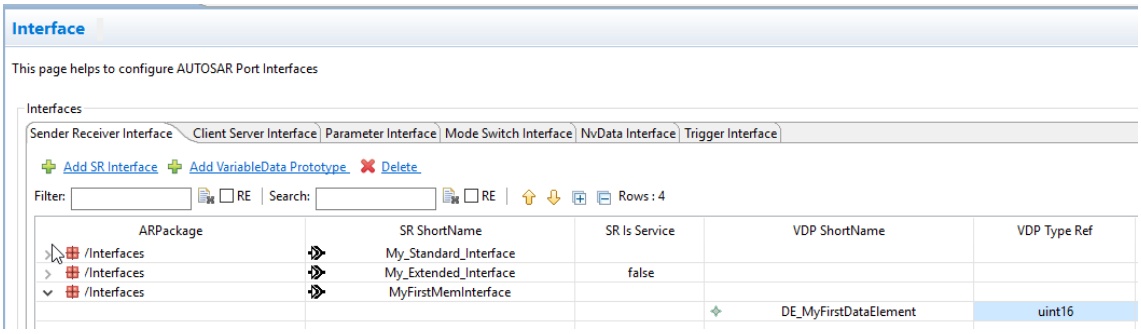


Place the Port Interface into the ARPackage named “Interfaces”.

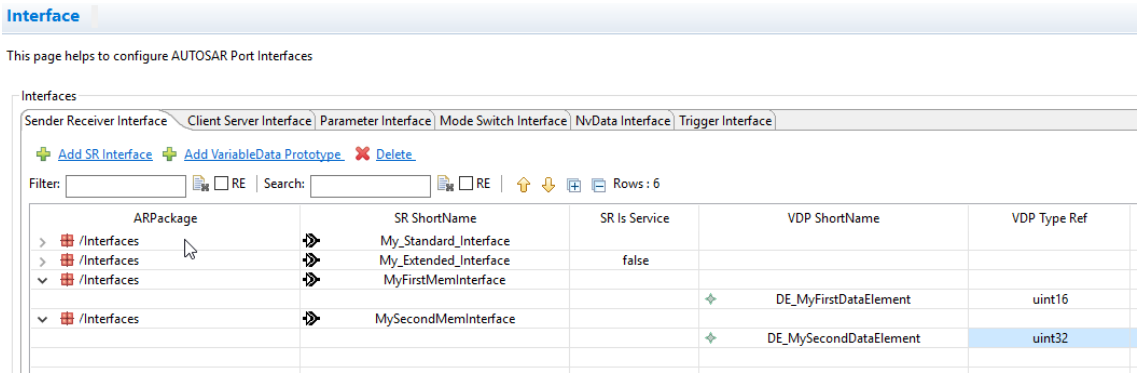
Double click on the Interface to open it with the “Data Dictionary Editor”; in “Sender Receiver Interface”, select the interface from the table and press the **Add VariableData Prototype** button.



Set the Data Element name and its Type reference. The result of this step is shown in Figure below.

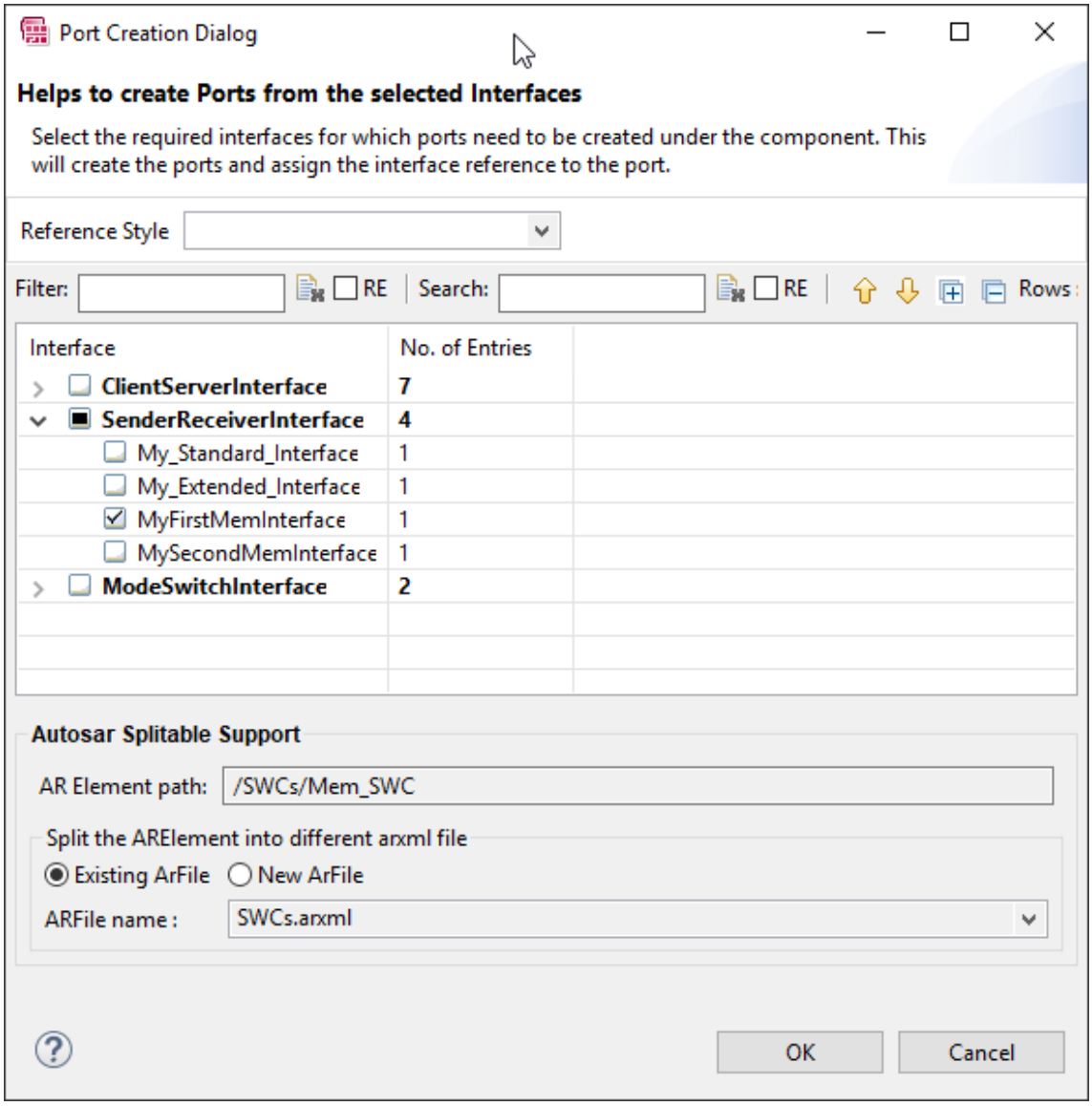


Create in the asme way a second interface for the second PIM; the final result of this step is shown below:

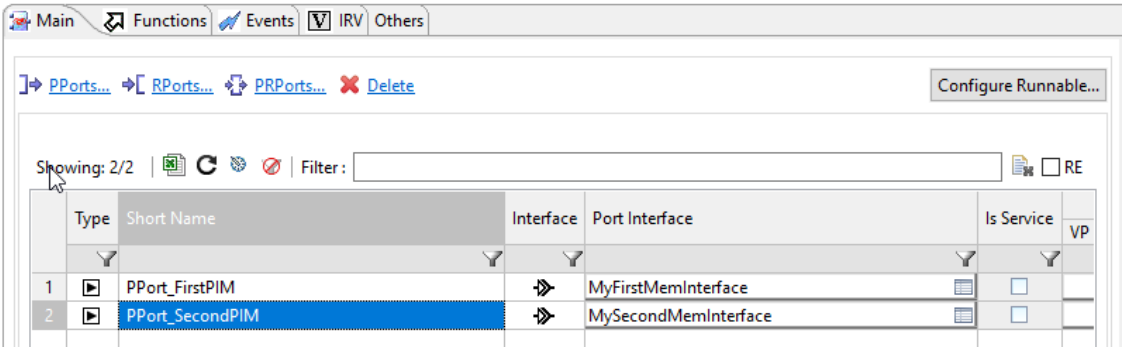


4.1.4 Create a new Port

SWCs communicate with each other through ports, the Memory SWC prototype has two provided port to eventually communicate datas to other SWCs and two RPorts to receive datas from NvM service module (created after configuation generation). To create a new port, open the SWC with “Component Editor”; in “Main” Tab press the **PPorts. . .** button. In the pop-up window “Port Creation Dialog”, select the desired Interface.

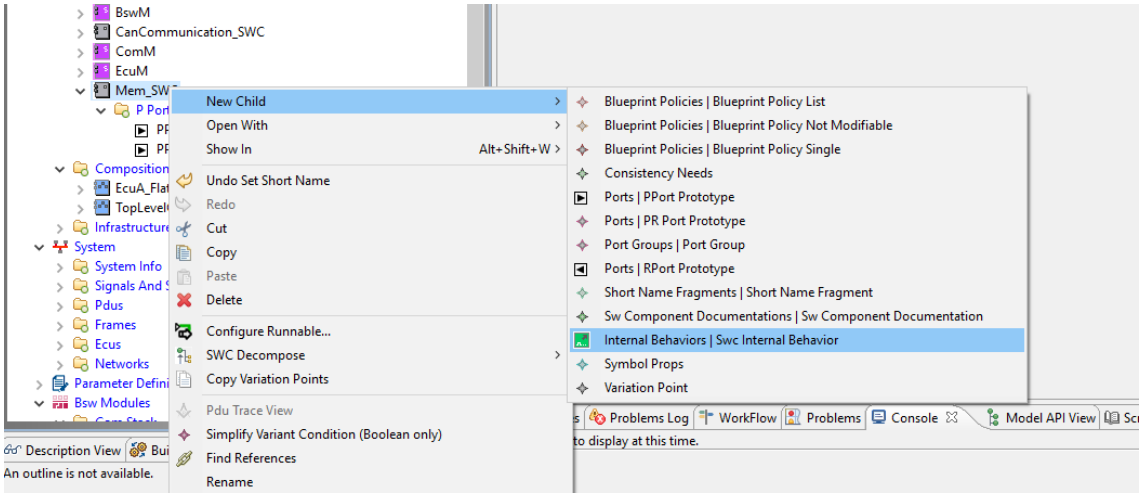


The final result of the Ports created is:



4.1.5 Create a new Internal Behaviour

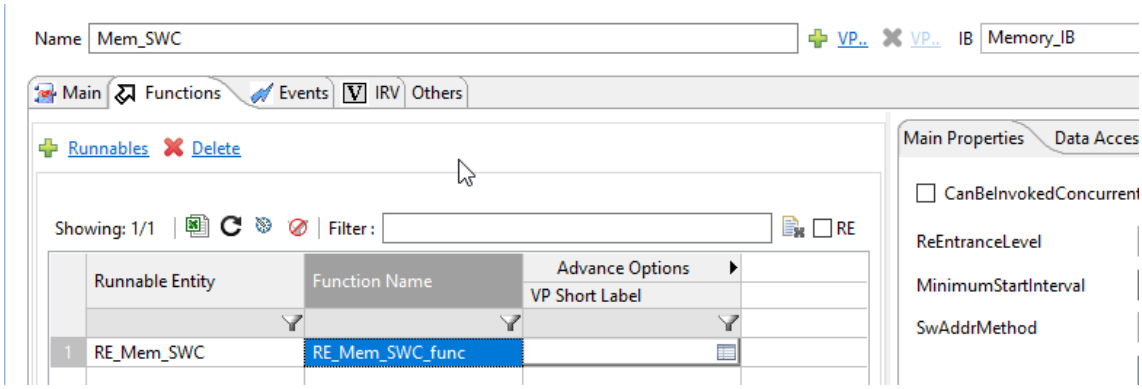
To fill the empty SWC an “Internal Behaviour” container must be created inside which create the entities such as runnables. To create the container, right click on the SWC and select **New Child –> Internal Behaviours | Swc Internal Behaviour**



Double click on the container to open it and set its properties, in particular set the nam.

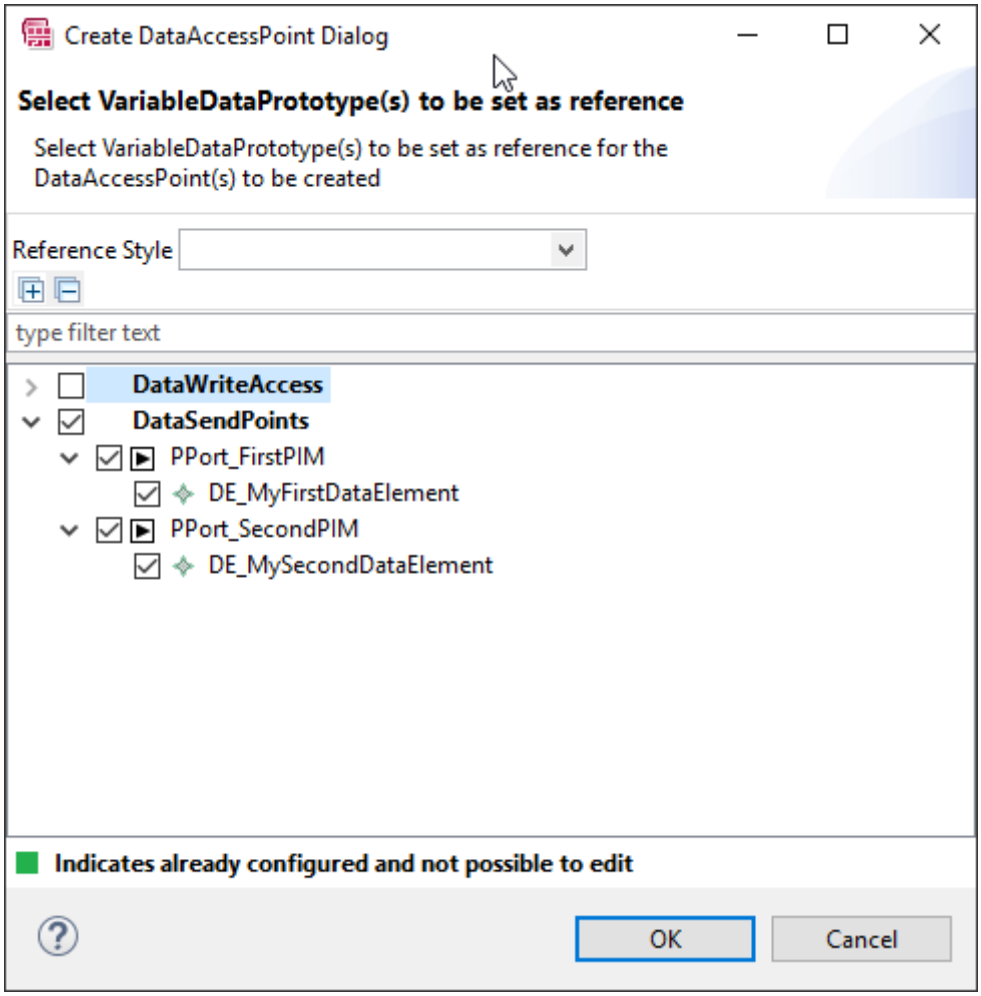
4.1.6 Create a new runnable

Now that the Internal Behaviour is created, you can create the runnables entities. To create a runnable, re-open the SWC prototype with the “Component Editor” and switch to the “Functions” Tab. Press the **Runnables** button and edit the runnable name and its Function name in the table.



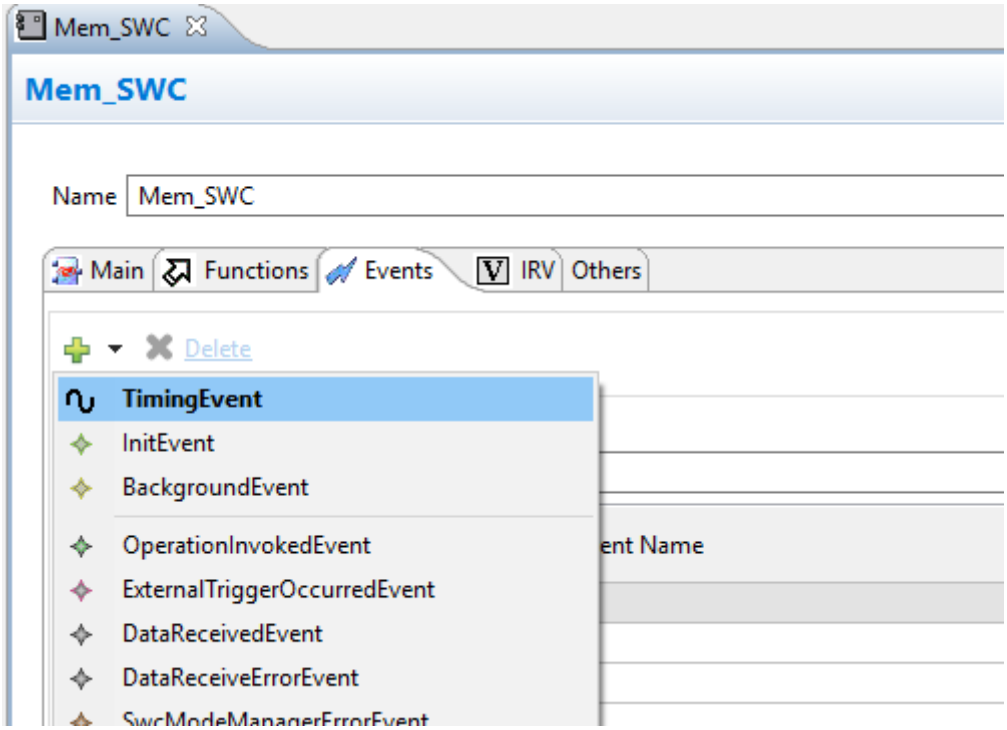
4.1.7 Create a new Data Access Point

Runnables have access to SWC ports only if a Data Access Point is created; to do so, in “Functions” Tab switch to the “Data Access Points” subtab and press the **Access Points. . .** button. In the pop-up window “Create DataAccessPoint Dialog” select the “Data Send Points”.

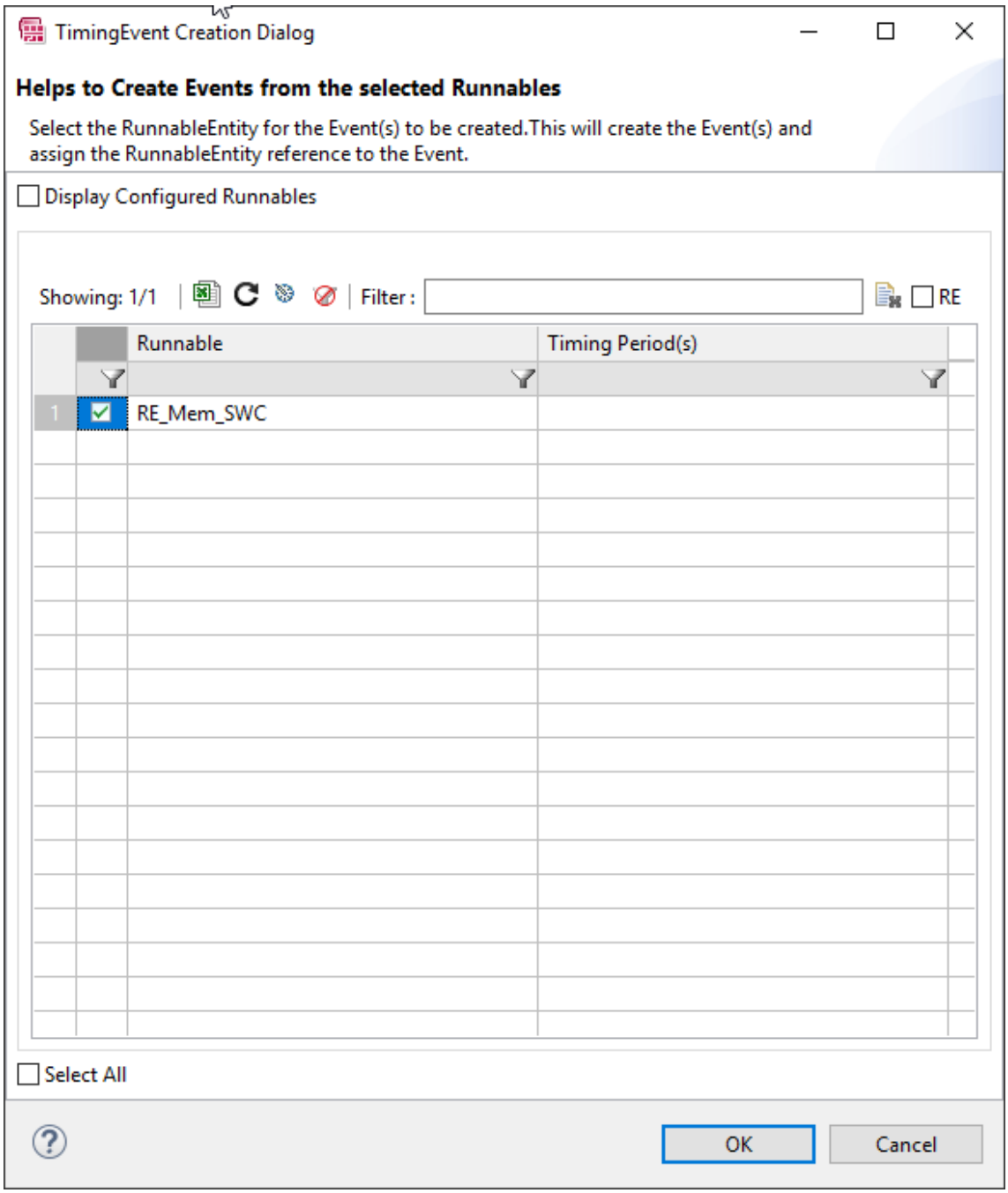


4.1.8 Create a new Event

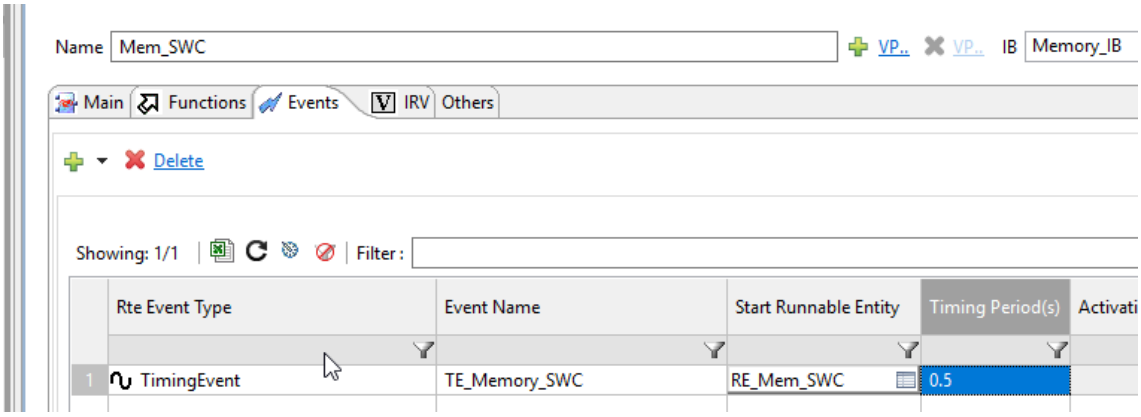
Runnables can be executed only if they are connected to an event. The event triggering the runnable is a timing event. To create the event switch to “Events” tab, press the little arrow next to the green plus and select “Timing Event”.



In the “TimingEvent Creation Dialog” select the desired runnable by a double click on it and press **Ok**.

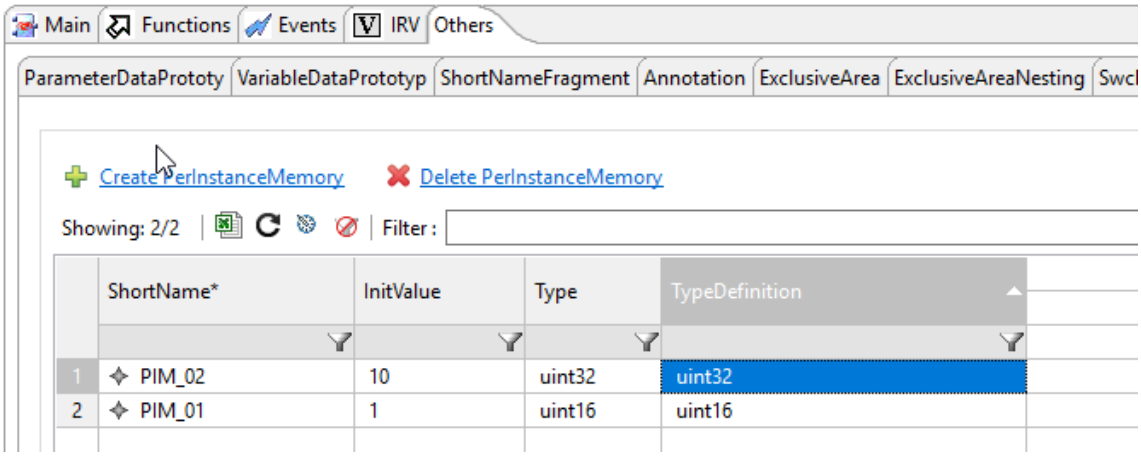


Configure the event's properties such as the the event name and the period, the result of this step is shown in Figure below.

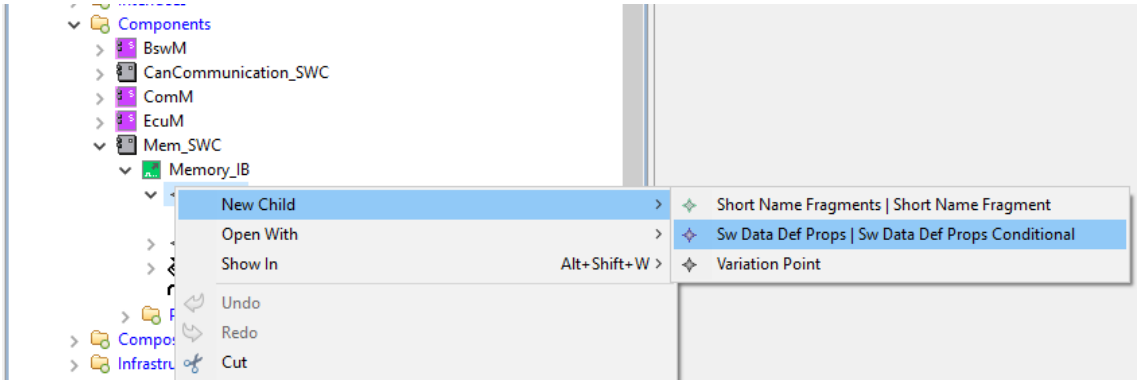


4.1.9 Crete the PIMs

To create a PIM from the Component Editor swithc to Tab “Others” and select the sub-tab “PerInstanceMemory”. Press the button **Create PerInstanceMemory**; a new element will be adde to the table below, edit the Shortname and set the initial value (InitValue column), type and type definition. The final result of the two PIMs needed is shown in the following figure:



For each PIM a new software data definition proposal must be created. To do that right click on the PIM and select **New Child –> Sw Data Def Props Variants | Sw Data Def Props Conditional**

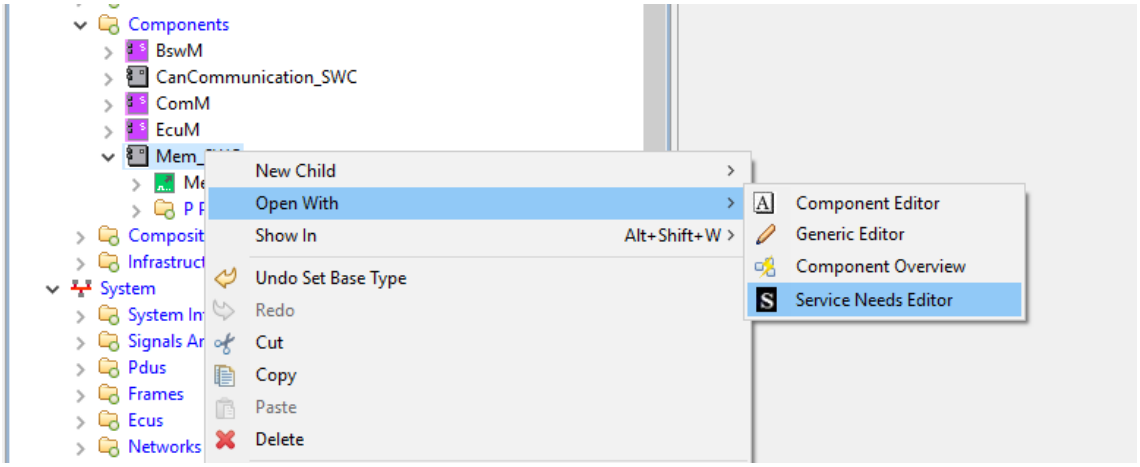


Open the created element and set the **BaseType** parameter to the proper value.

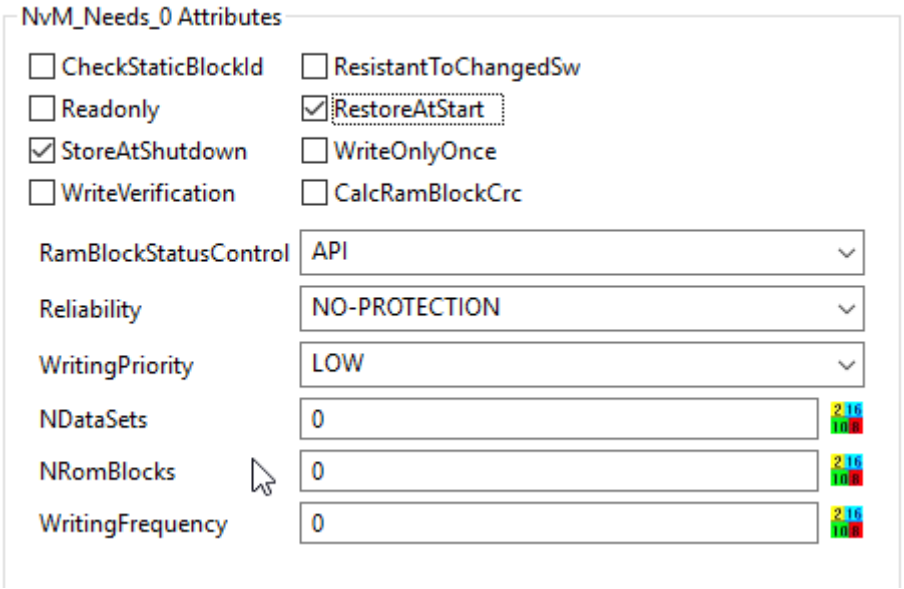
Do the same for all the PIMs configured.

4.1.10 Create NvM Service Needs

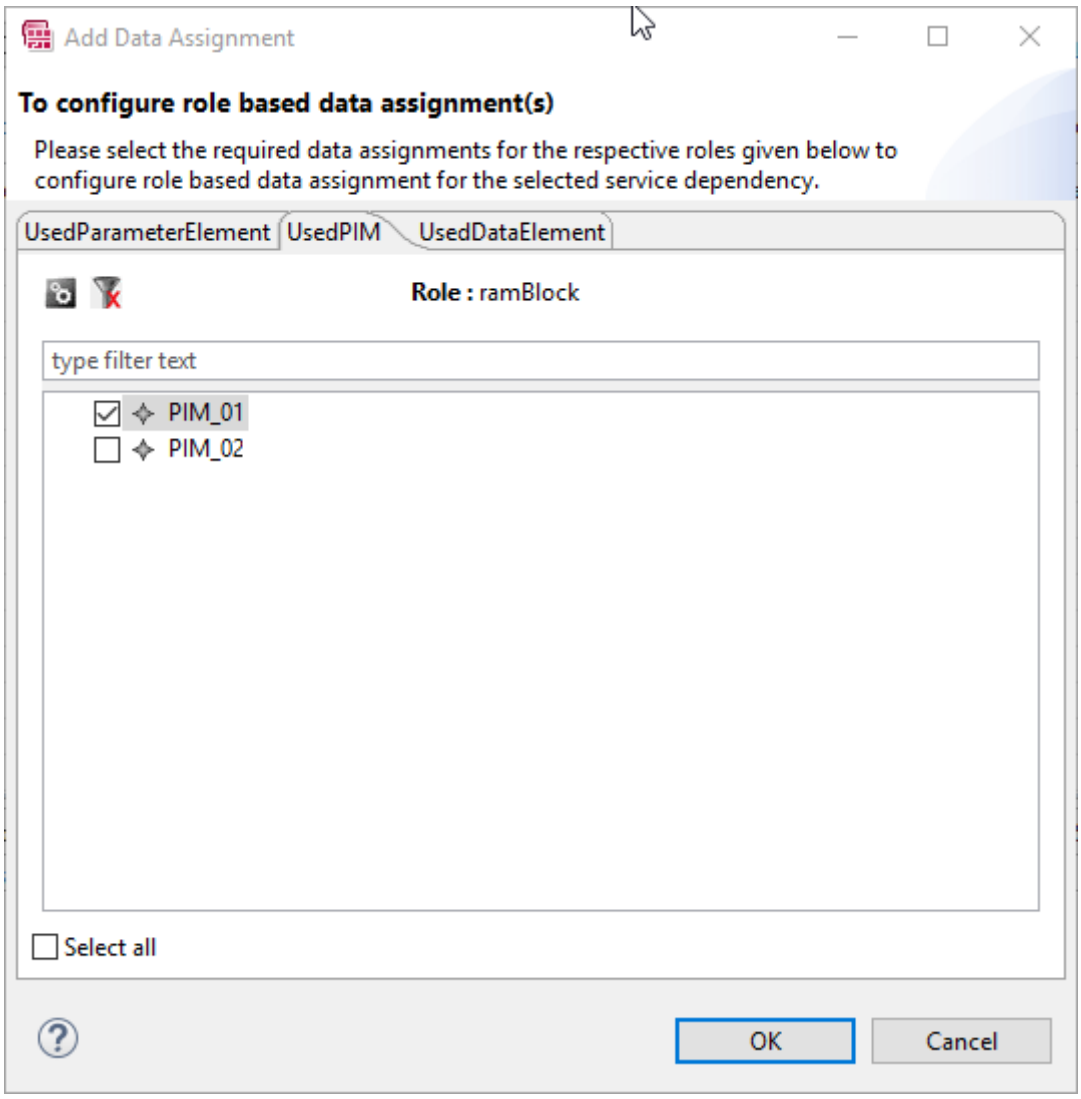
Open the SWC containing the PIMs with the service needs Editor; to do that right click on the SWC and select **Open with –> Service Needs Editor**.



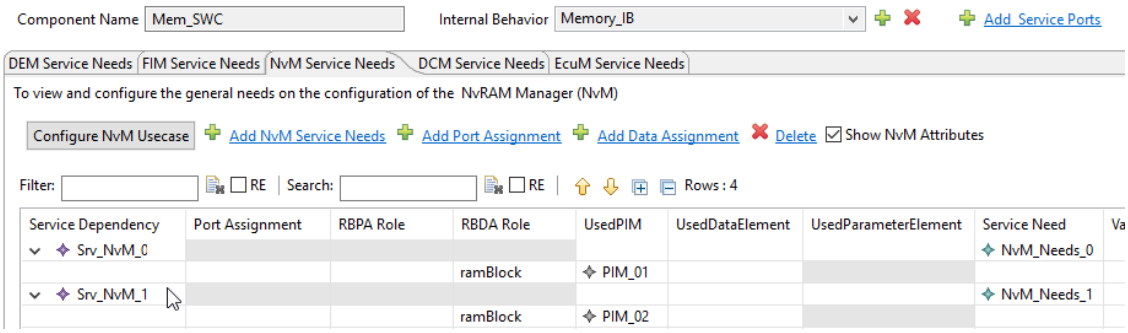
Switch to Tab **NvM Service Needs** and select **Add NvM Service Needs**; the table below will be filled with a new element named “Srv_NvM_0”; edit the name and the service needs attributes, below an example of how to configure the attributes to have a PIM that is read at each startup and written at each shutdown.



Finally connect the PIM the the Service needs created by adding a data Assignment; press on **Add Data Assignment**; in the po-up window “Add Data Assignment” switch to tab **Used PIM** and select the desired PIM; then press **OK**:

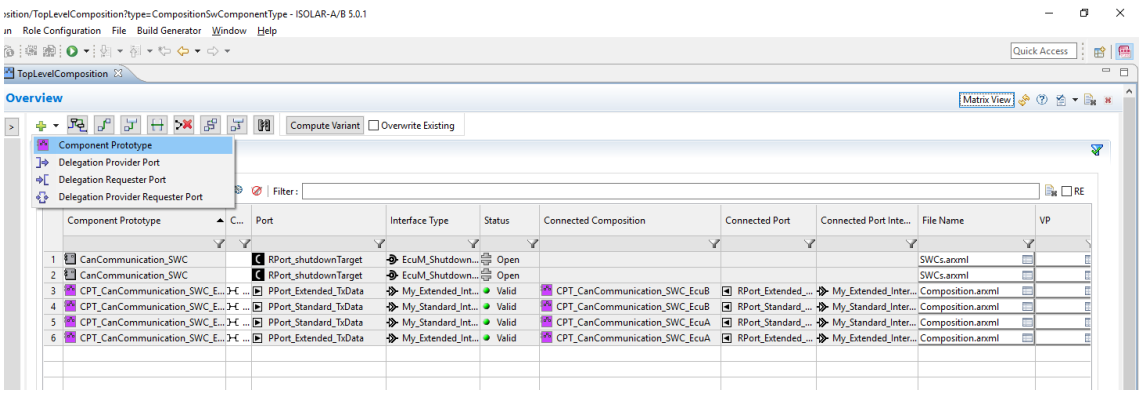


Do the same steps for all the PIMs; the final result should be as shown here:



4.1.11 Update System Composition

Open the TopLevelComposition and press the little arrow next to the green plus button and select **Component Prototype**.



In the pop-up window “Create Component Prototype Dialog” select the new ASW SWC and press **Ok**.

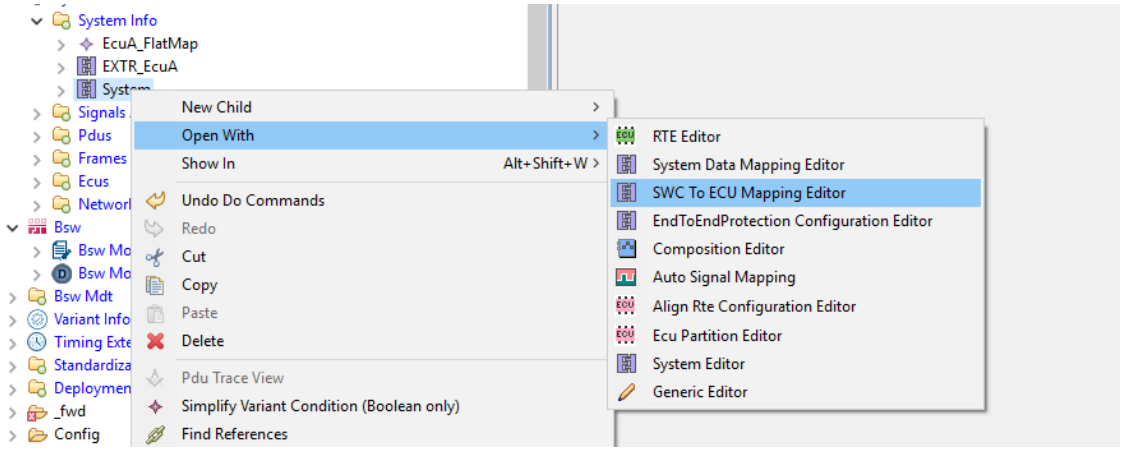
4.2 Part 2 - ECU Configuration

In this part ISOLAR B is used to configure the BSW modules of the ECU.

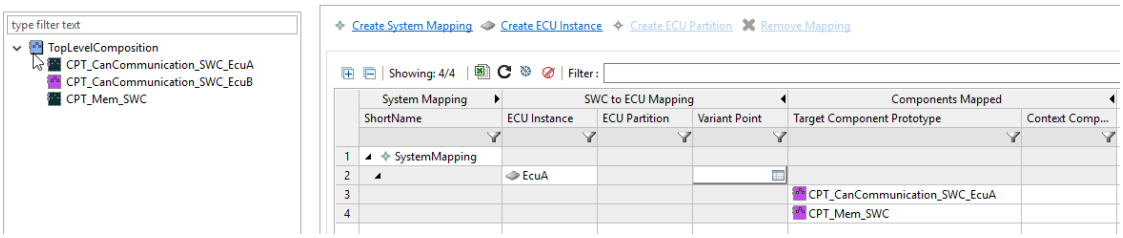
NB: ISOLAR B can be accessed by selecting the “ECU Navigator” in the Project Explorer view.

4.2.1 Map SWC to ECU

The new SWC must be mapped to the ECU “EcuA”; to do that right click on the System element (in the imported project it is called **System**) and select **Open with –> SWC to ECU Mapping Editor**



On the right the TopLevelComposition should have in black the mapped SWC and in pink the not mapped ones. Drag and drop the Memory SWC under the EcuA on the left; the final result of this step should be as shown below:



4.2.2 Update ECU Extract

The existing ECU Extract must updated now since a new SWC has been assigned to the ECU; right click on the System and select **Create ECU Extract** in the pop-up window that appears keep selected the “Update existing ECUExtract” option and press **Finish**.

NB: if the previous ECU extract files are not placed in the project root folder, this step will create two new files; replace the old files with these new ones moving them in the desired sub-directory.

4.2.3 Configuration Generation

Now you can update the Configuration of the BSW generating a new one.

Press the ConfGen button or from the Menu bar select **RTA-BSW –> Automatically configure BSW from System Description**. This step will update the “Config” directory inside the Project folder with the automatically generated arxml files related to the BSW modules. As a result expanding the Bsw Modules within the “ECU Navigator” menu you can see how a Memory stack has been created with the modules: NvM, MemIf and Fee.

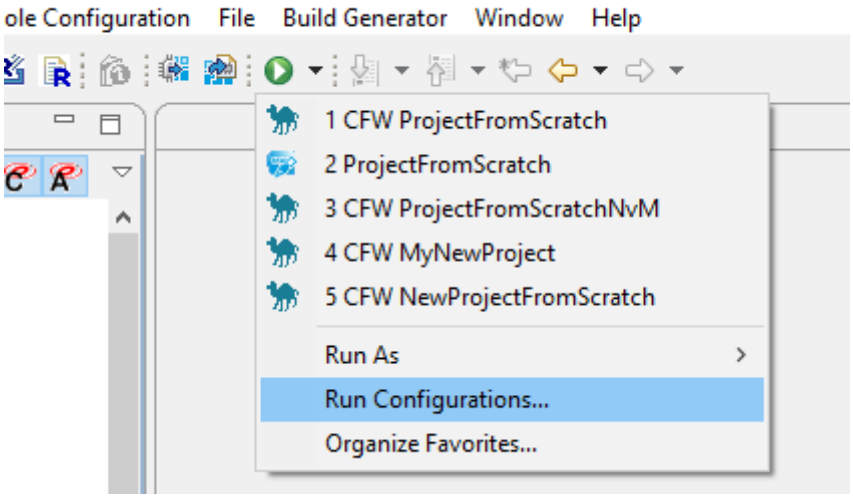
4.2.4 Import configuration

Import from the MCAL the Fls module description files; in this case the following files have been imported:

- Fls_BSWMD.arxml
- Fls_BSWMD_EcucValues.arxml

4.2.5 Code generation

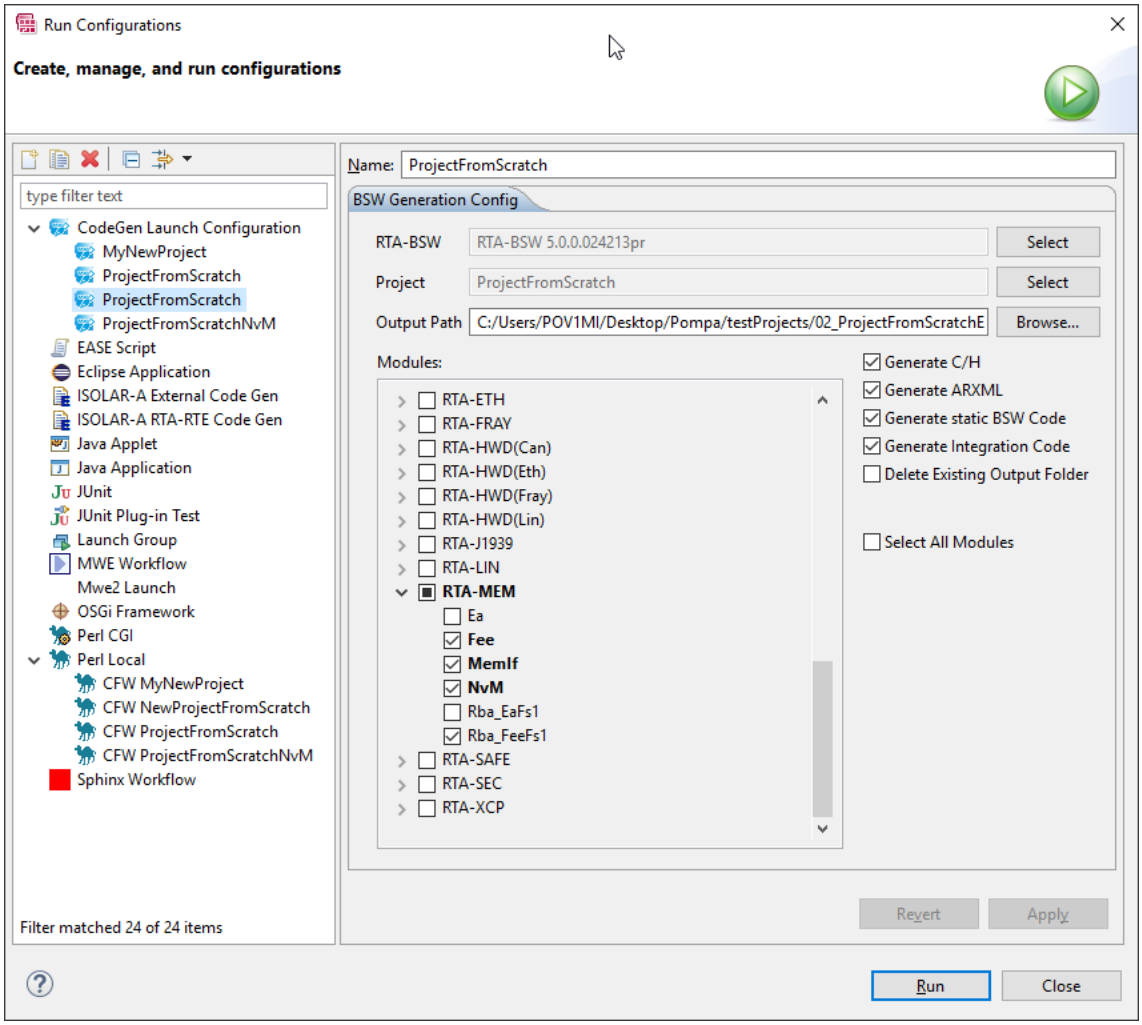
Now you need to generate the BSW code; in this way the BSWMD files will be generated and the NvM service will be recognized and available to be added to the composition. First you need to have a configuration to run. From the Toolbar press the arrow next to **Run Configurations** button and select **Run Configurations. . .**



In the pop-up window select the correct configuration

Make sure that the RTA-BSW version recognised is the desired one and that the project is correct, check the path where you want to create the folder with the BSW source code. Finally check that

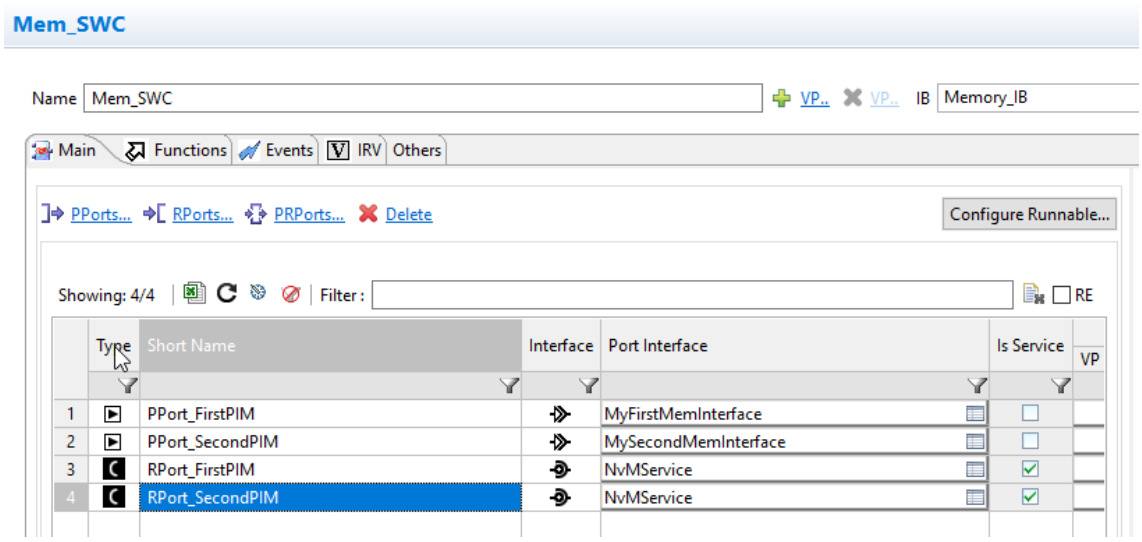
all the BSW modules configured are checked. The result of this step is shown in Figure below.



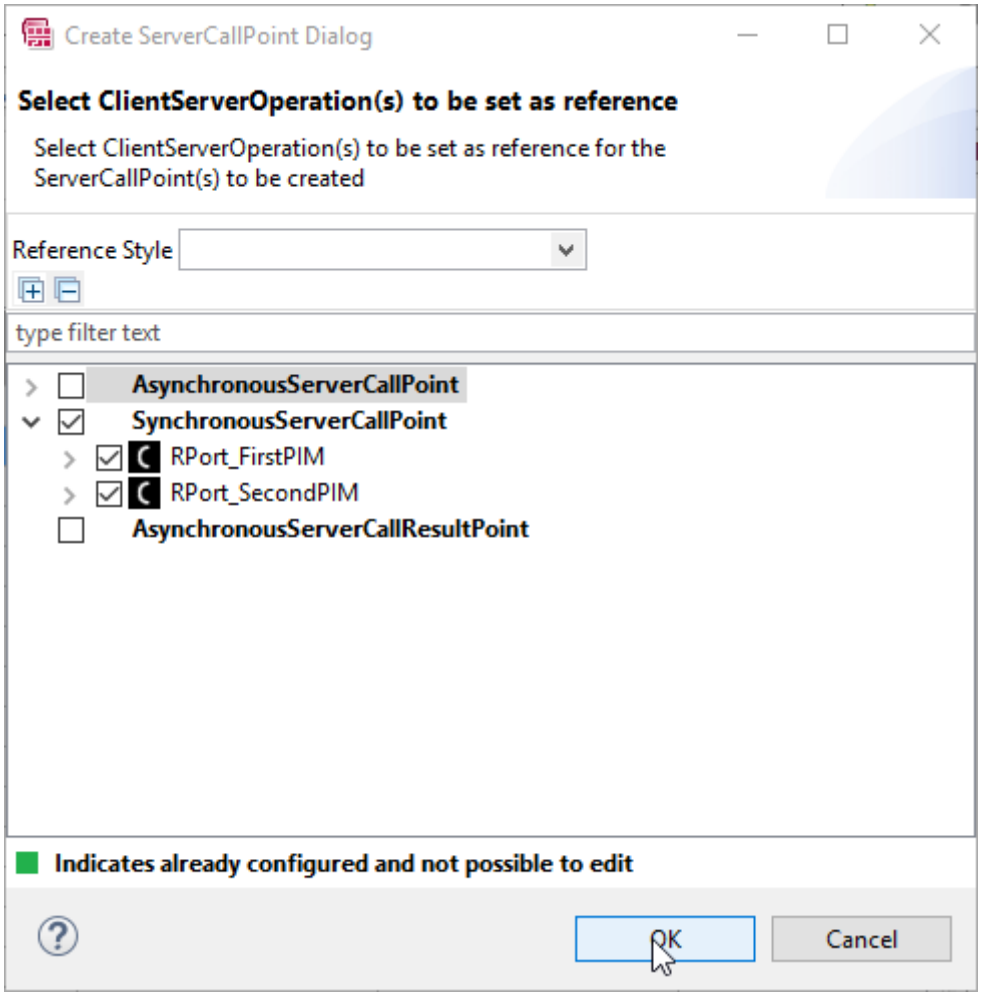
Make sure to add the needed code into Integration files generated since the existing files will be overwritten! (e.g. in Compiler.h fill the INLINE defines)

4.2.6 SWC Ports for NvM

Besides the PPort created previously, the SWC “Mem_SWC” needs to have as many RPorts as are the PIMs; so open the SWC with the Component Editor, switch to tab “Main” and create a new RPort; in the pop-up window select **ClientServerInterface –> NvMService**. Do the same for every PIM created.

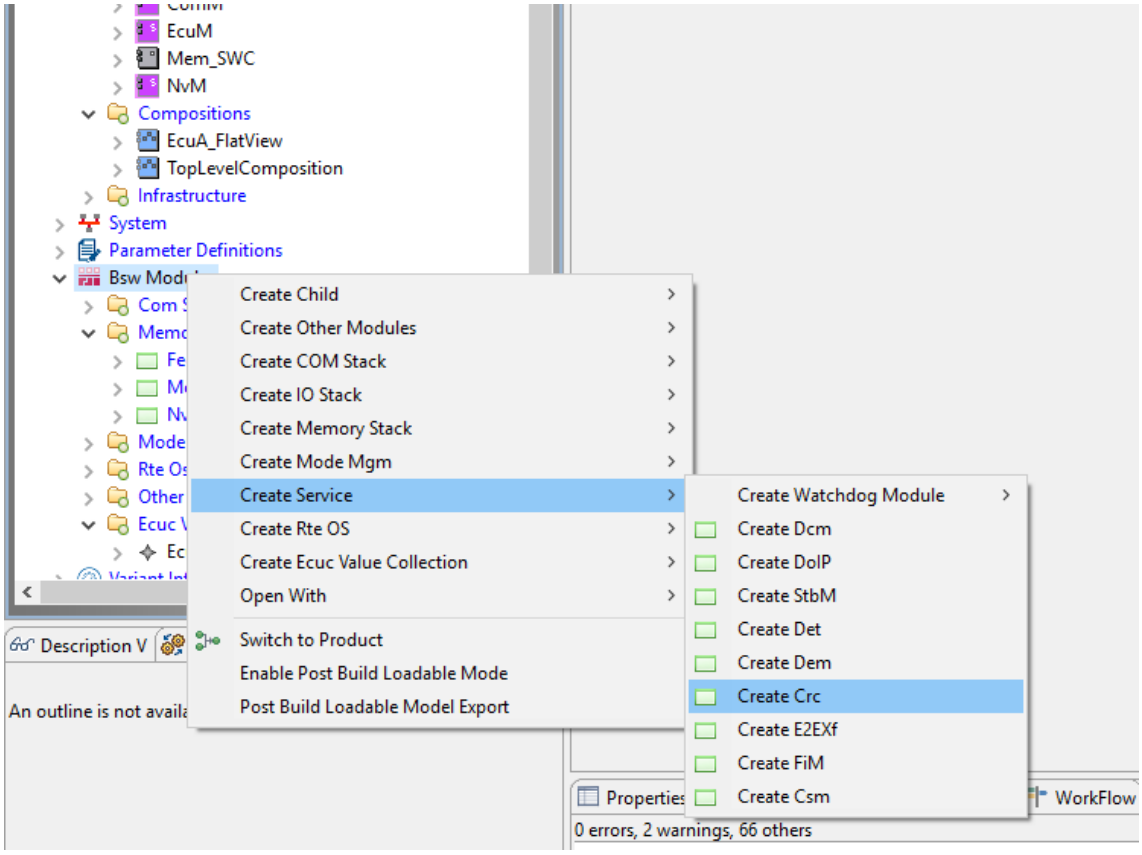


Reopen the SWC with the component editor to add the Server Call Points; switch to “Functions” Tab, on the right switch to sub-tab “Server Call Points” and press the **Server Call Points. . .** button. In the pop-up window select “SynchronousServerCallPoint”; in this way needed Server Call Points for all the PIMs will be created; see figure below:



4.2.7 Crc Module creation

By defualt the NvM automatically configured by ConfGen will make use of the CRC; the corresponding BSW module must be configured. To do that from the **ECU Navigator** menu right click on **Bsw Modules** and select **Create Service –> Create Crc**. In the pop-up window choose the module name “Crc” and press **Finish**.

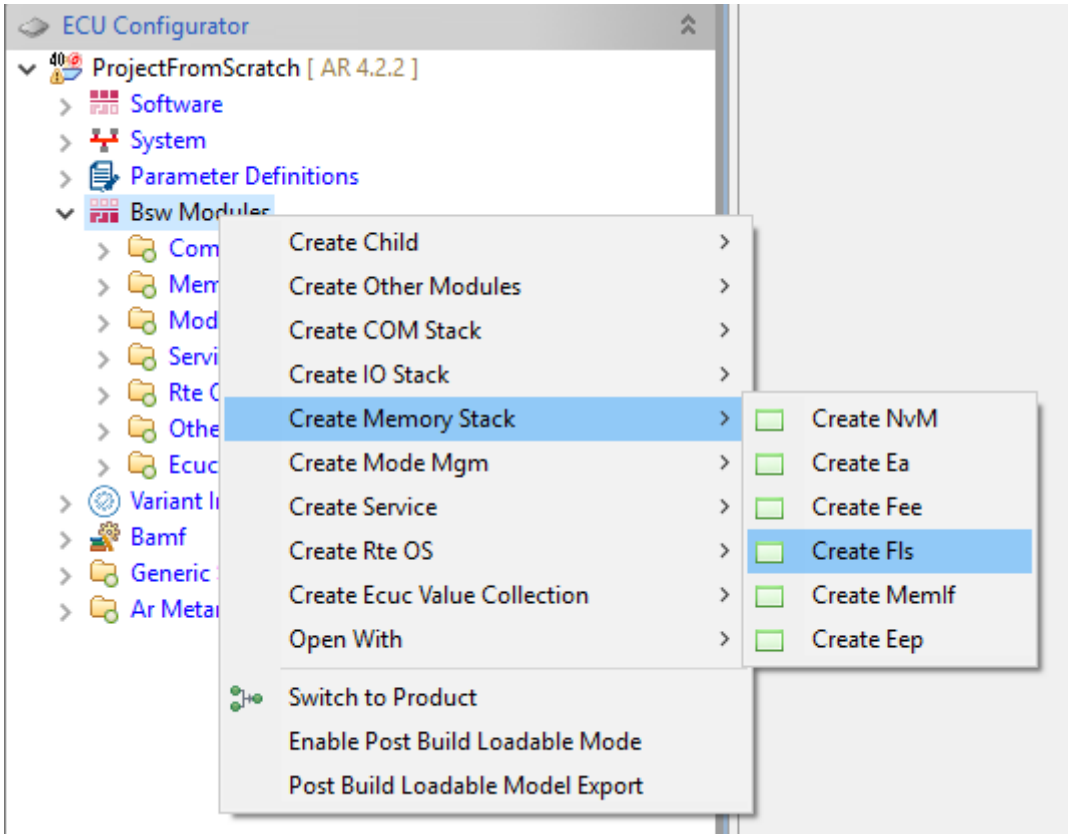


Remember to configure the Crc module according to the project’s requirements.

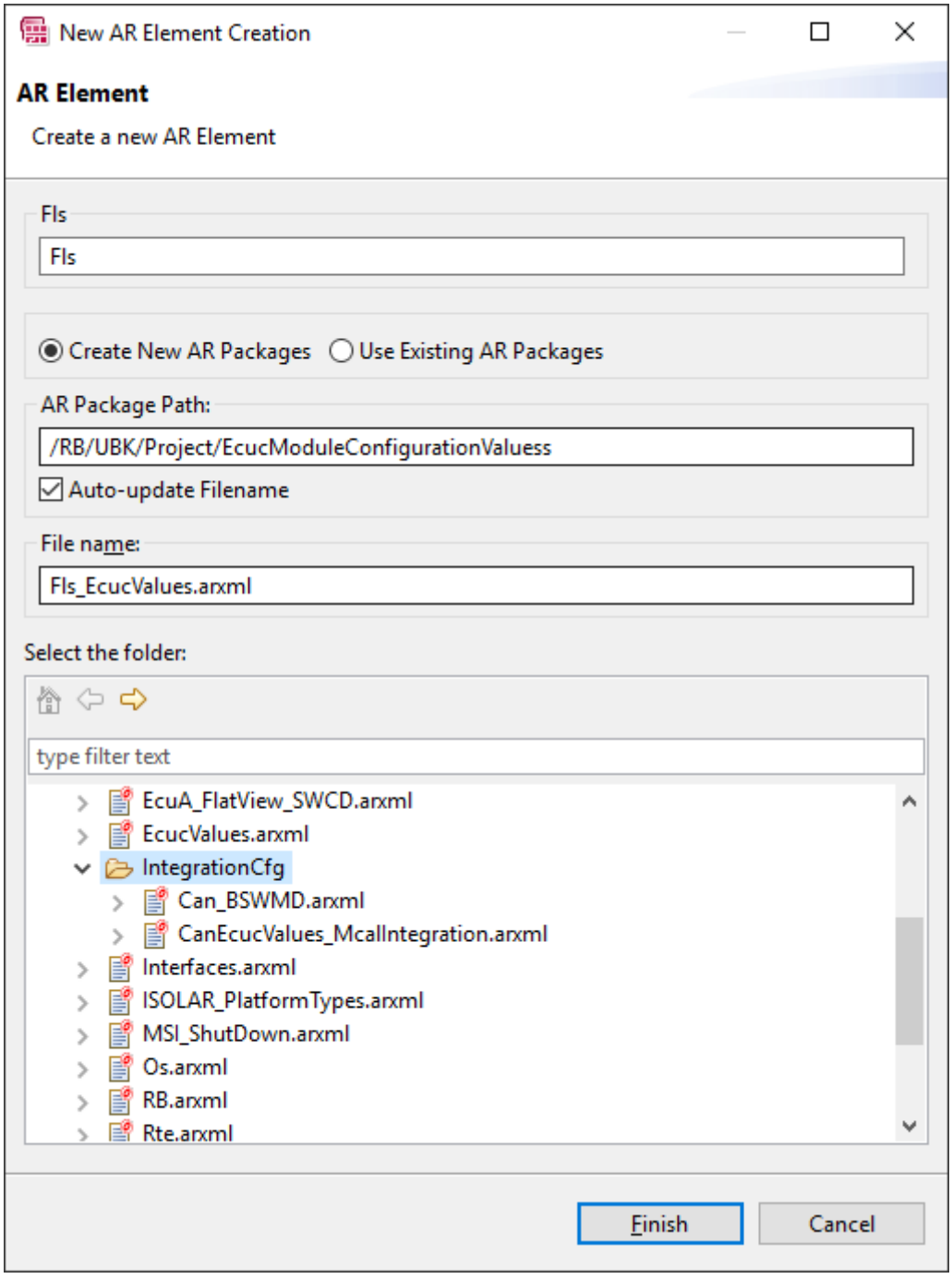
After this step it is necessary to run the code generation again to generate this module’s code.

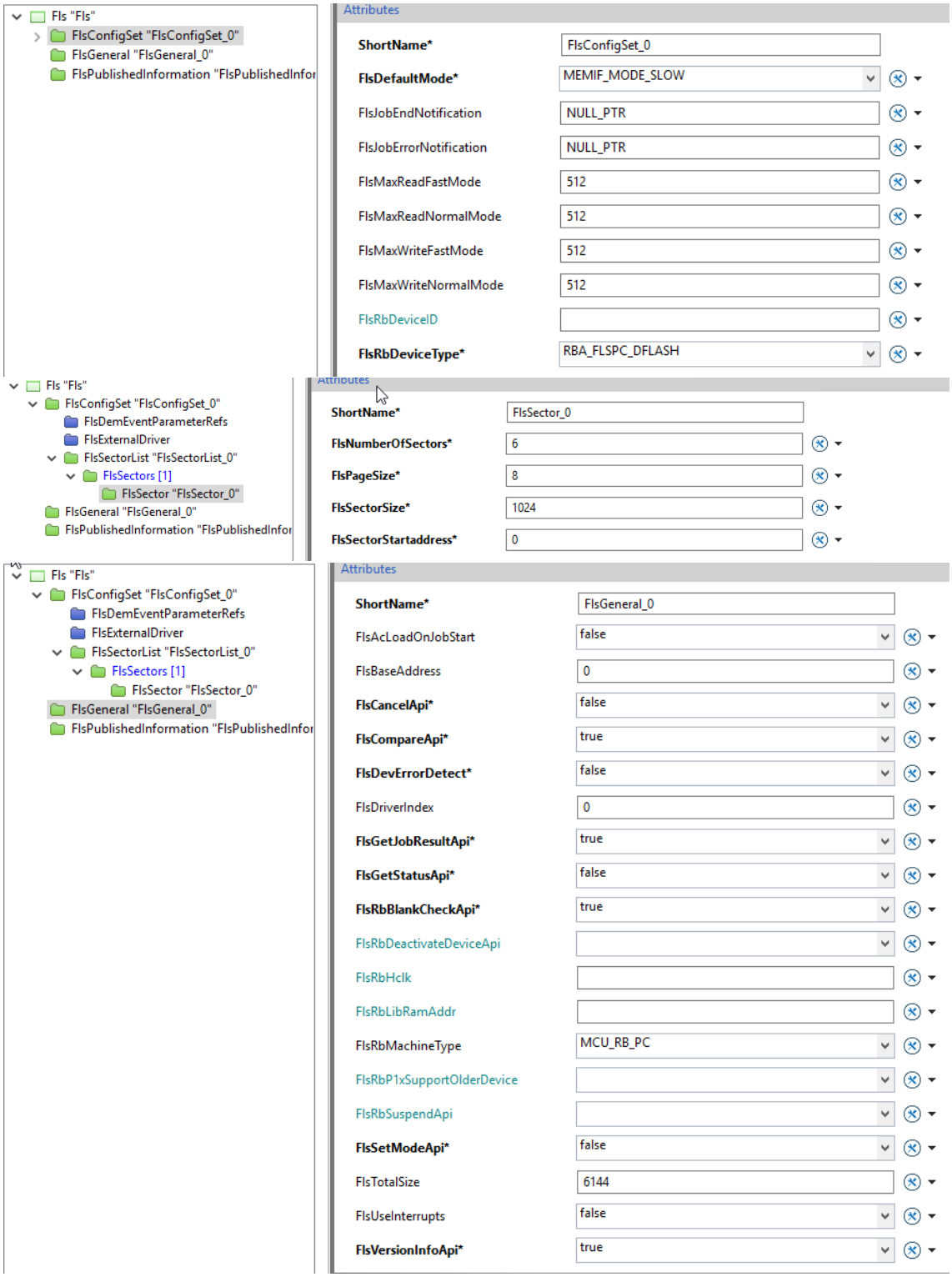
4.2.8 Fls module creation

Create an module “Fls” to have all the reference needed to the MCAL. To create an “Fls” module right click on “Bsw Modules” from the “ECU Navigator” menu and select **Create Memory Stack –> Create Fls**



Configure the module according to you hardware; reported below a sample configuration to be used with virtaul target:

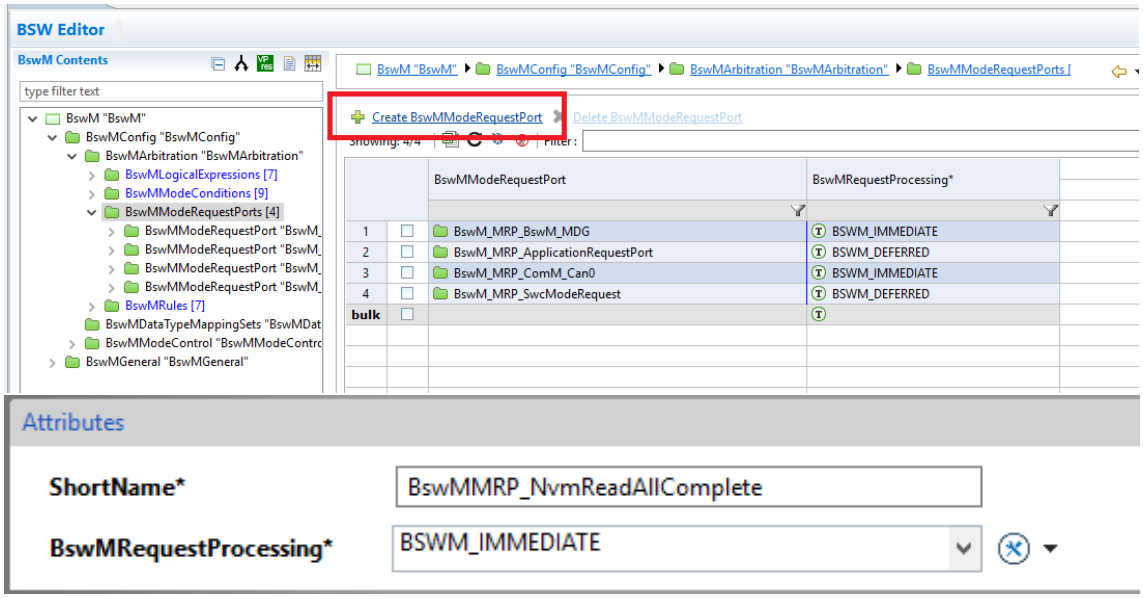

New AR Element Creation
AR Element
Create a new AR Element
Fls
Fls
Create New AR Packages
Use Existing AR Packages
AR Package Path:
/RB/UBK/Project/EcucModuleConfigurationValuess
Auto-update Filename
File name:
Fls_EcucValues.arxml
Select the folder:
type filter text
EcuA_FlatView_SWCD.arxml
EcucValues.arxml
IntegrationCfg
Can_BSWMD.arxml
CanEcucValues_McalIntegration.arxml
Interfaces.arxml
ISOLAR_PlatformTypes.arxml
MSI_ShutDown.arxml
Os.arxml
RB.arxml
Rte.arxml
Finish
Cancel



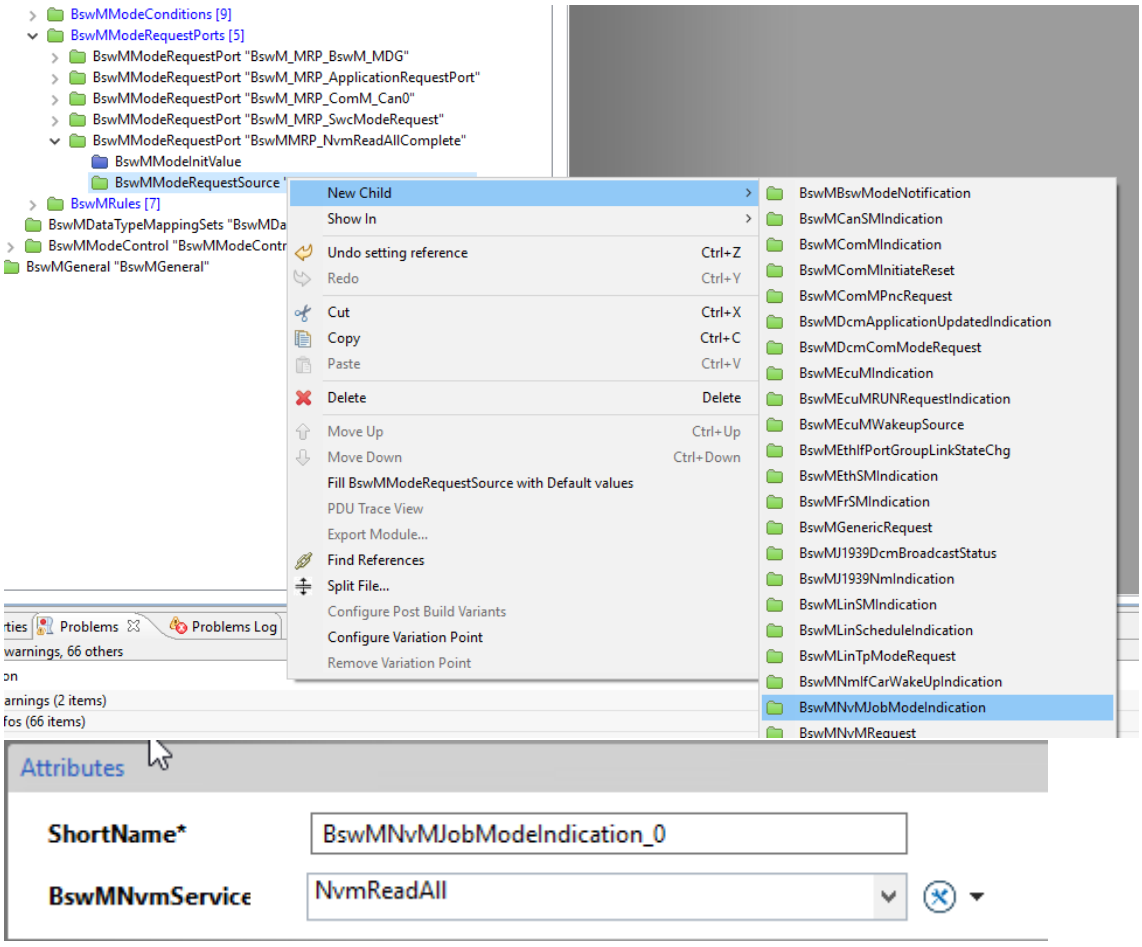
4.2.9 Edit BswM module

Memory stack needs to be initialized at startup and deinitialized at shutdown, for this reason the BswM module needs to be modified in the following way:

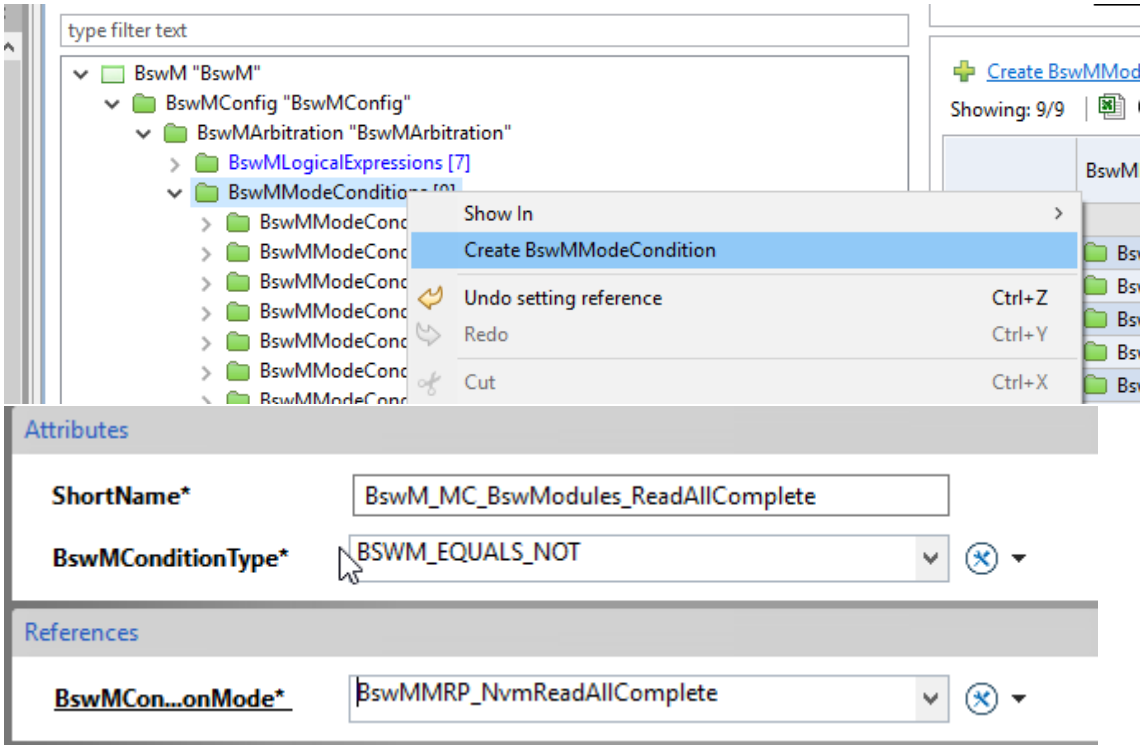
1. Add a new Mode request port with request processing of type **BSWM_IMMEDIATE**



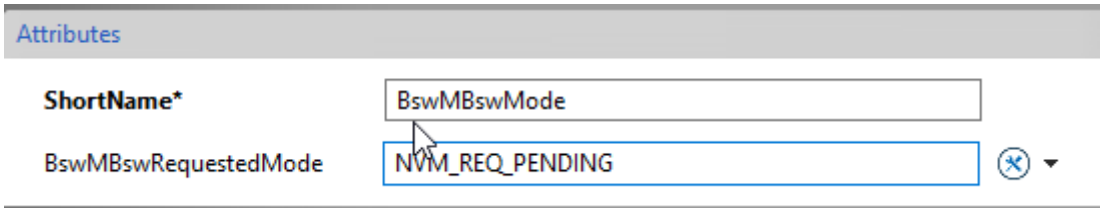
2. Set the MRP source right clicking on the source and selecting **New Child –> BswMNvMJobModeIndication**; then set the **BswMNvmService** to “NvmReadAll”.



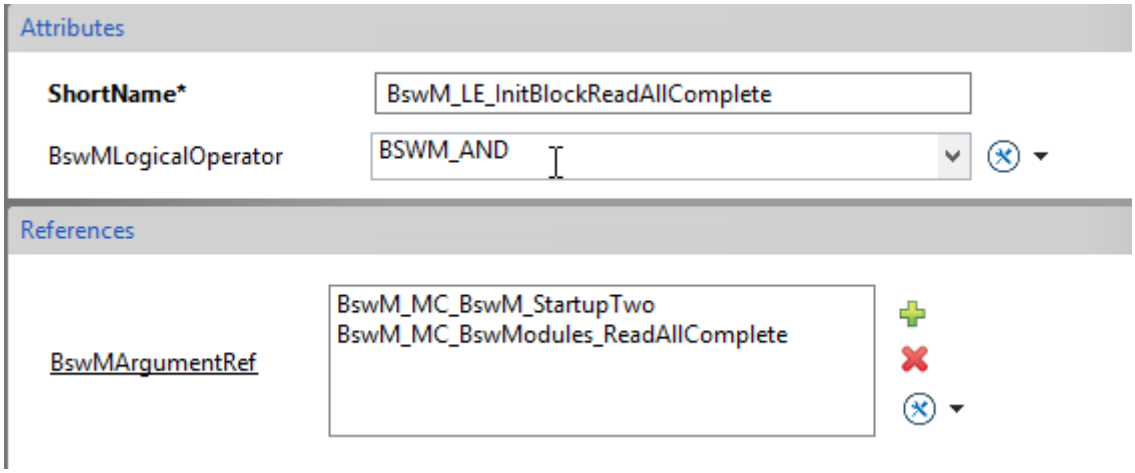
3. Create a new Mode Condition to check the NvM reading at startup at set its parameters as below:



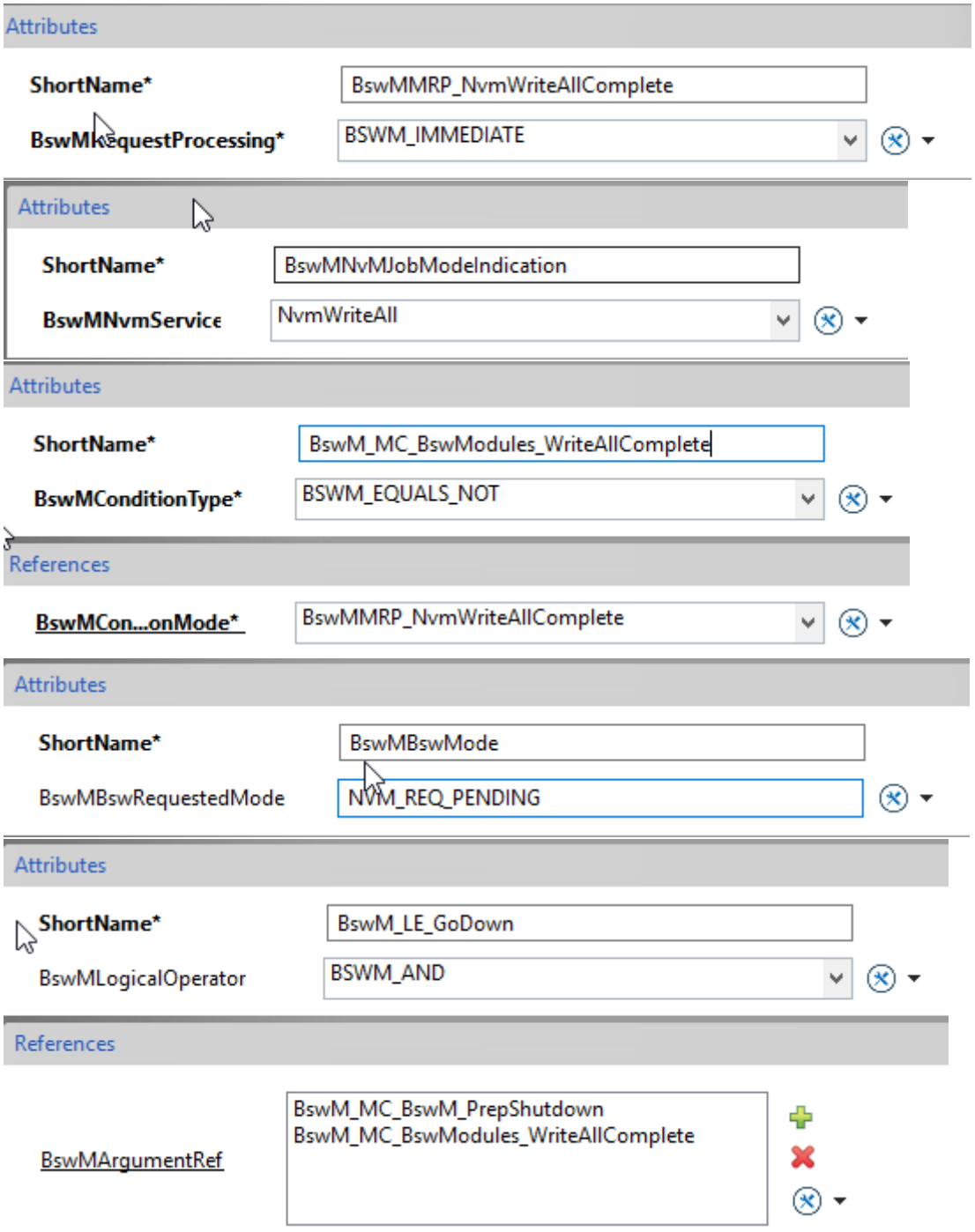
Finally set the BswMConditionValue to:



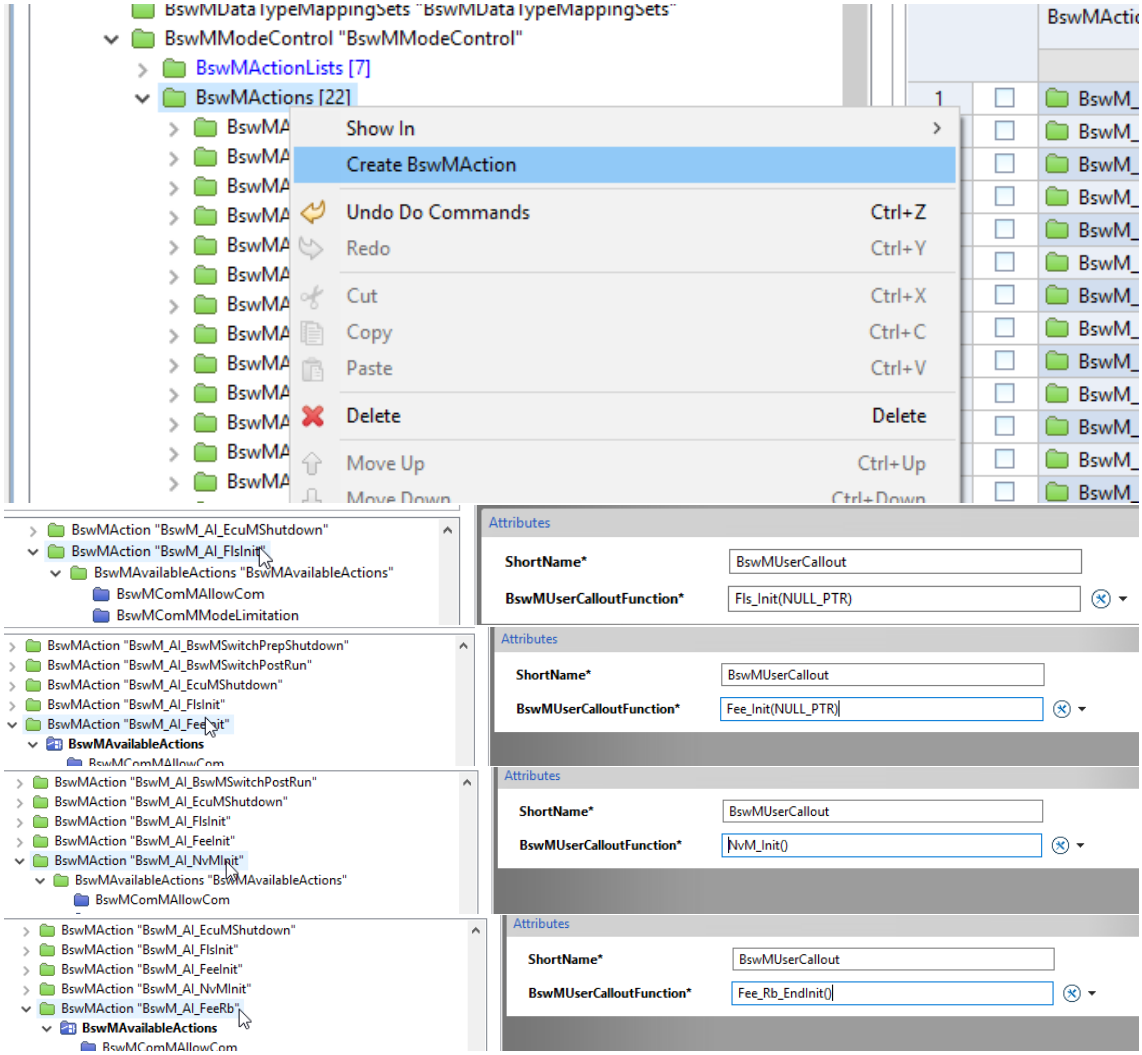
4. Edit the Logical Expression “BswM_LE_InitBlockReadAllComplete” to add the created MC. The final configuration of this Logical Expression must be as shown below:



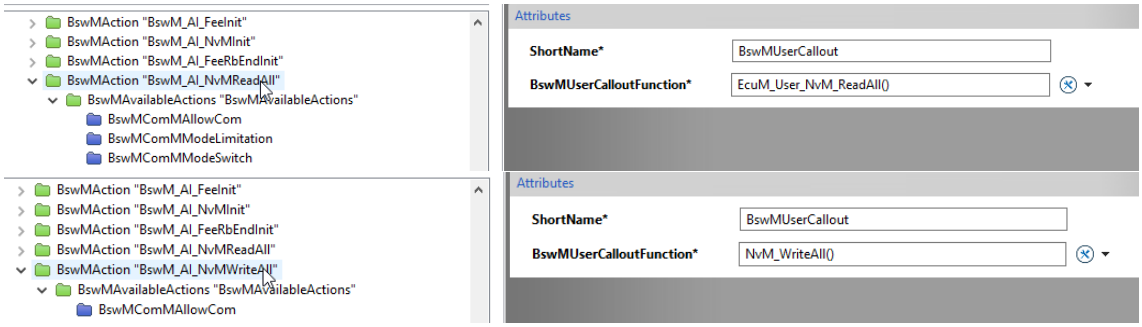
What has been done in steps 1 to 4 for the startup reading, must be done for the writing at the shutdown. Elements must be added or edited, see the figures below as reference:



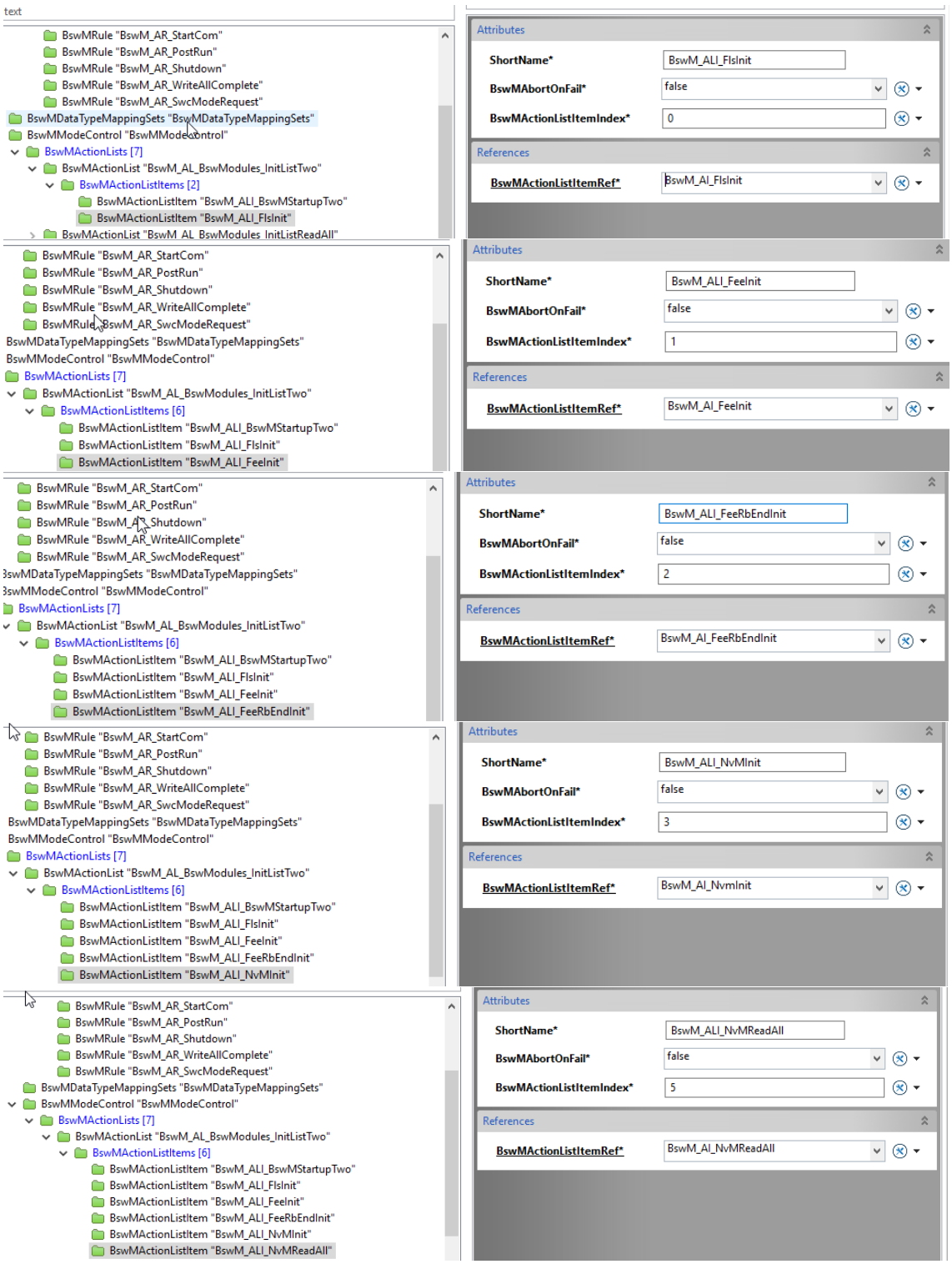
5. Add the Memory stack modules initialization functions to the Actions; in the following pictures an example on configuration:



6. Add the Actions for reading and writing of NvM:

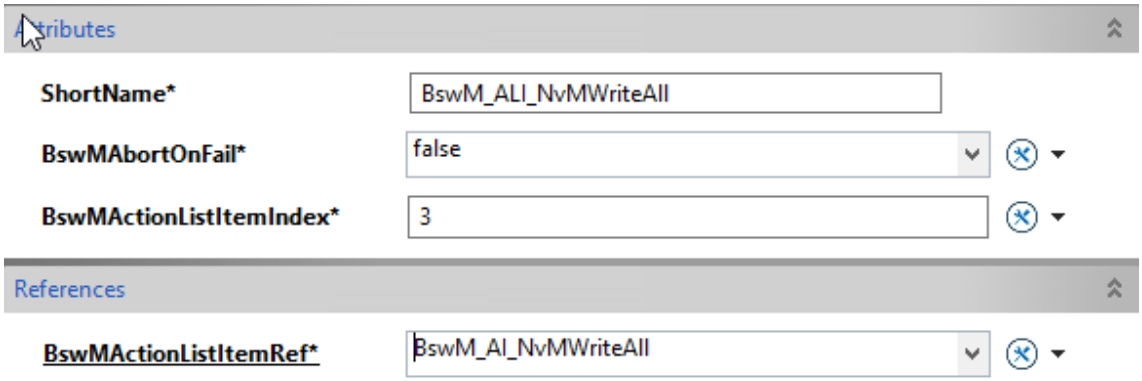


7. Edit the Action List “BswM_AL_BswModules_InitListTwo” to add the modules initialization and the read all action:



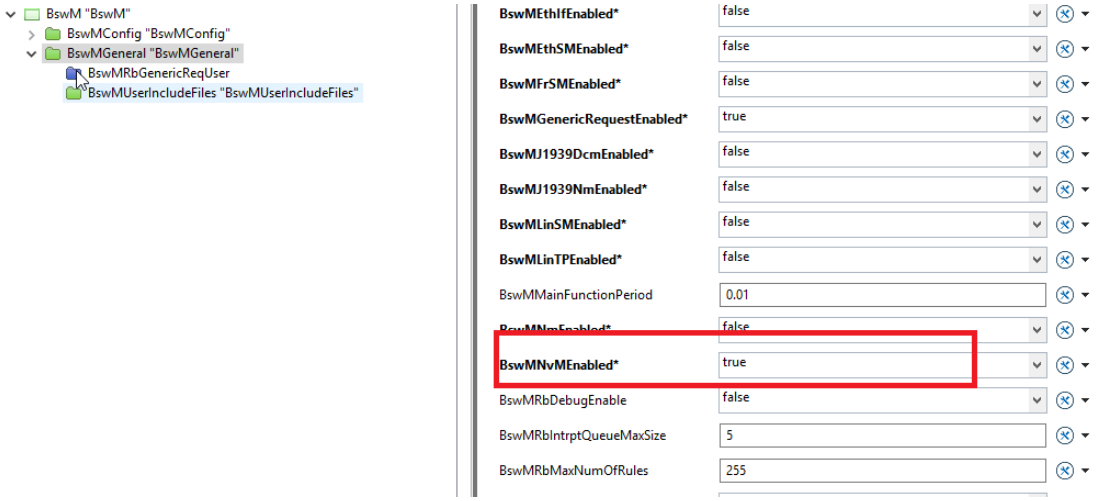
NB: the action InitBlockTwo must have an index such that it is executed after the memory stack modules initialization and before the read all function!

8. Edit the Action List “BswM_AL_Shutdown” to add the writing of NvM action:



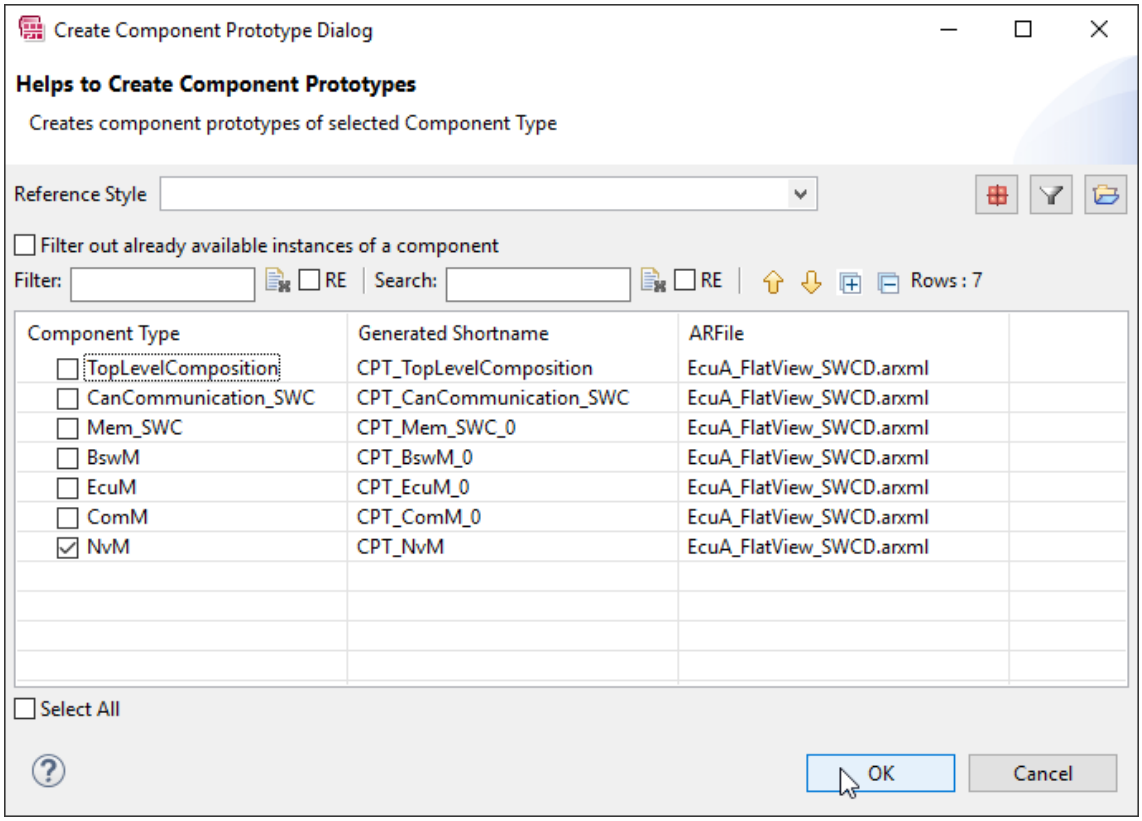
NB: The writing action must be executed ater stopping the RTE, deinitializing the other modules and while the ECU is in PREP_SHUTDOWN state

9. Edit the “BswMGeneral” container parameters to enable the NvM:



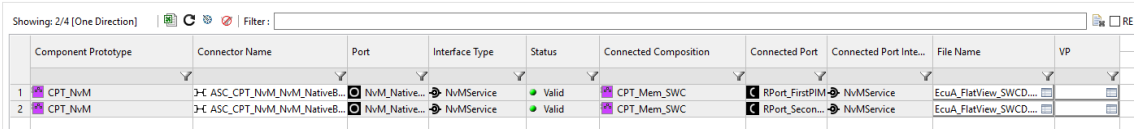
4.2.10 Composition Update

Now that the NvM service has been created you can add it to the Composition; open it with the Composition Editor, click on the little arrow next to the green plus button and select **Component Prototype**. In the pop-up window select the NvM component and press **Ok**



4.2.11 Create connections between SWC and NvM module

Now that the the NvM has been added to the ECU configuration, you must connect its ports to the SWC created previously. To do that open composition and switch to tab “Manual Connection Editor”. On the left select the NvM module and select the port “NvM_NativeBlock_2”; it must be connected to the SWC so on the right chhose the memory SWC and select the corresponding port. Do the same for the other Nvm Block. The final result of this connection step il shown below:

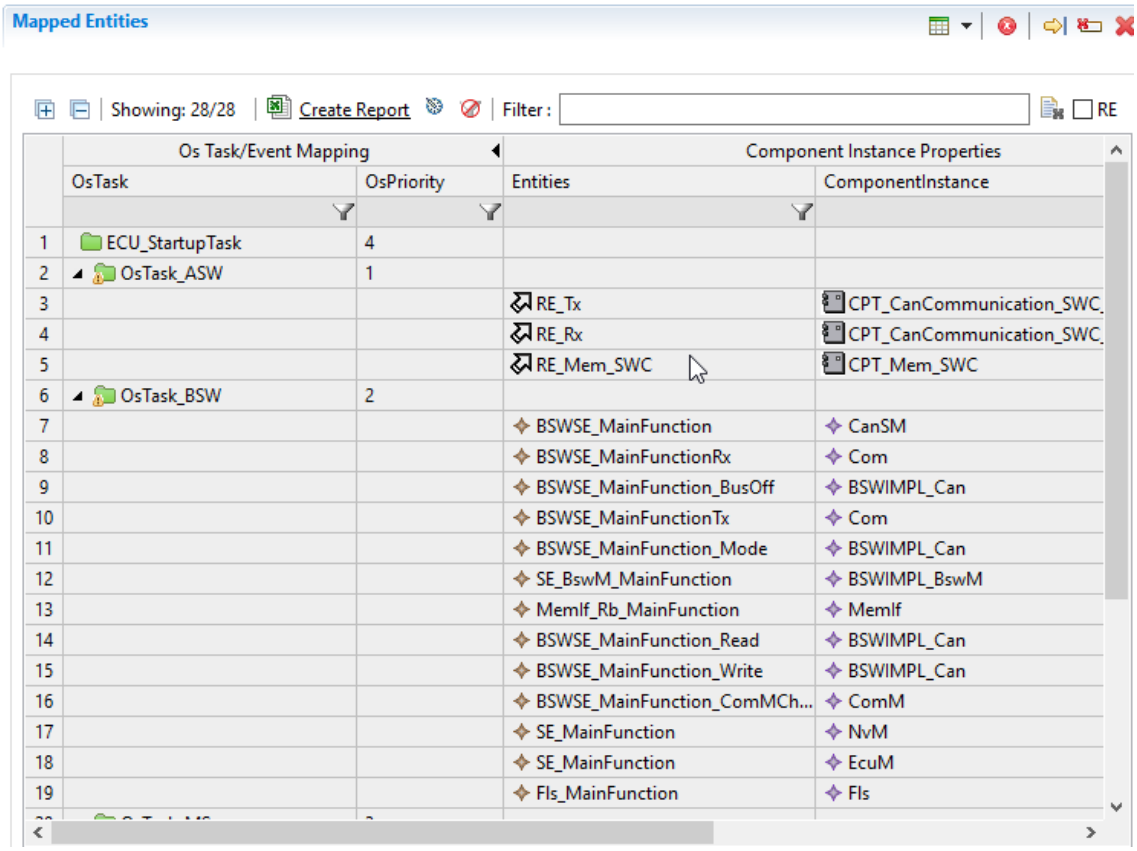


4.2.12 Update ECU Extract

Update the ECU Extract since a new module has been added. Right lcick on the System and select **Create ECU Extract** as done before.

4.2.13 Map the SWC runnables on Os Tasks

Open the existing Ecuc value collection and switch to the “Entity to Task Mapping” Tab and drag and drop the present runnables (SWC runnables, MemIf Main function runnable, NvM Main function runnable and Fls Main function runnable) on the right under the Os task on the left. The result of this step is shown in Figure below.



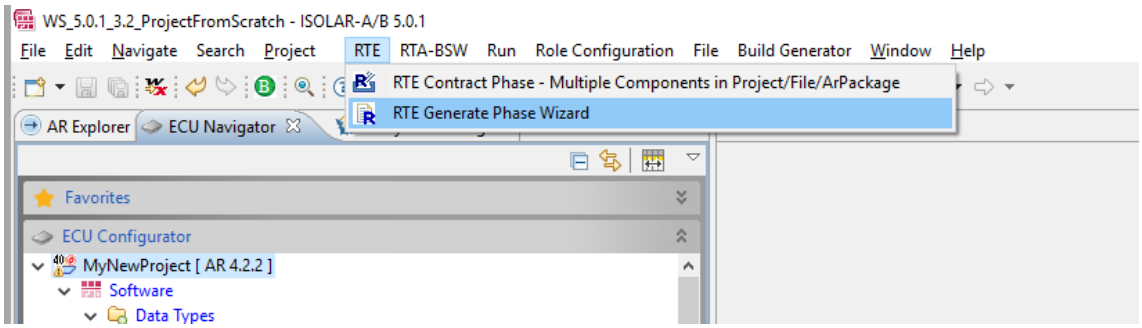
4.2.14 Code generation

Redo the code generation step to update the code with the current configuration.

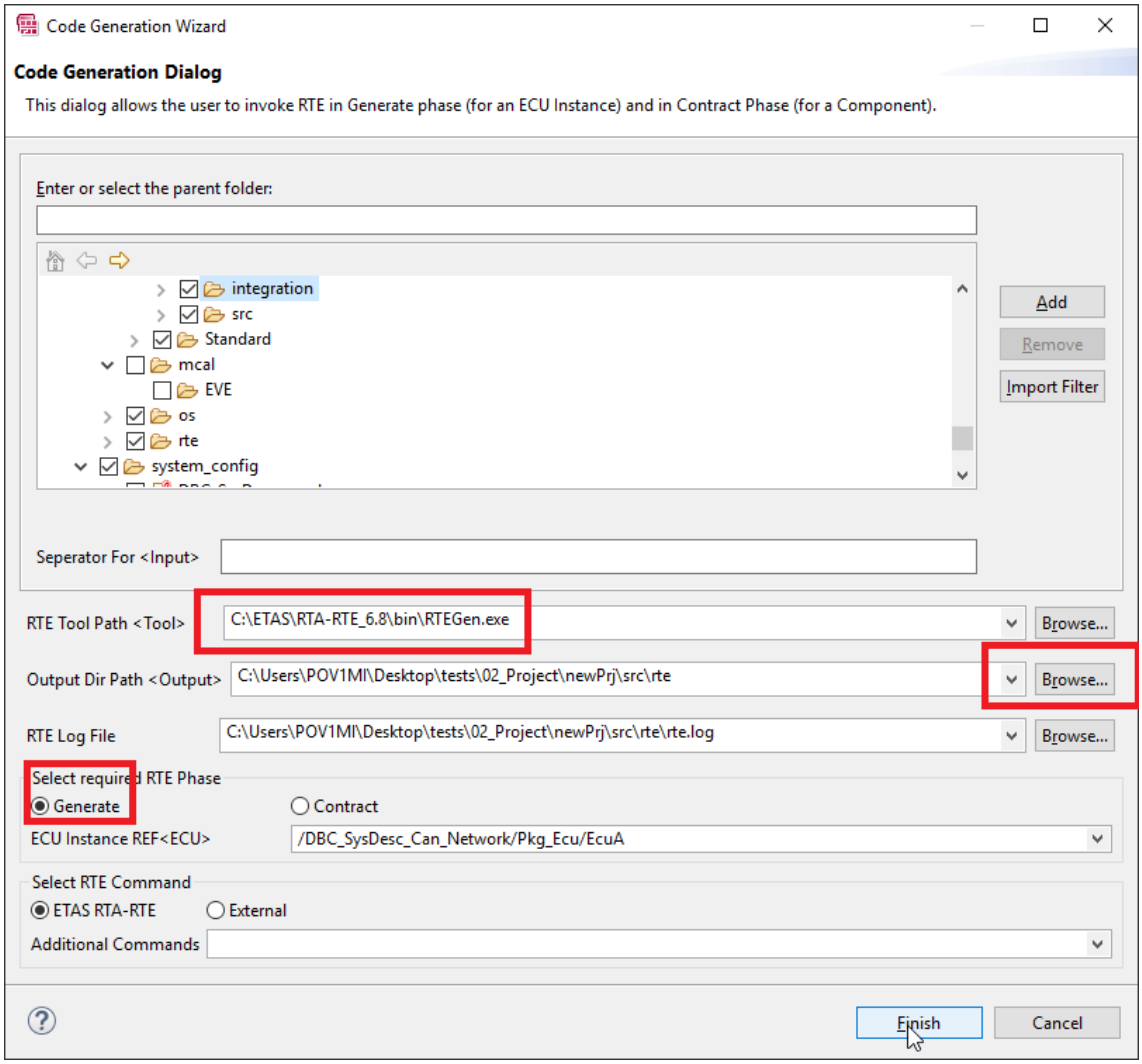
4.3 Part 4 - RTE

4.3.1 RTE generation

You can now generate the RTE code. To do this, use the RTA-RTE plug-in, press the **RTE** button in the menu bar and select **RTE Generate Phase Wizard** from the drop down menu.



In the pop-up window make sure to select the correct paths to the RTA-RTE tool and to the output folders. In the Additional commands always add the command (*`--os-define-osenv=RTAOS40`)* to create the necessary define for the AUTOSAR release used. If there is any port of BSW unconnected and you want to generate the RTE leaving them as they are, use the additional command (*`--strict-unconnected-rport-check=warn`)*.



NB: if the virtual target is used and the mcal configuration files are inside the project folder, make sure to deselect them in the RTE generation dialog window.

4.4 Part 5 - MCAL update

Remeber to update the MCAL. This step is target dependent; in this AN the virtual target is used so the MCAL is regenerated using the mcalgen.exe tool. In particular go to the project directory “ecu_config–>mcal” and edit the batch file to add the memory configuration file (Fls_EcucValues.arxml)

4.5 Part 6 - ASW

Do not forget to edit the SWC file adding the necessary code to run the test on the NvM; it means having variables to put read NvM values into and other variables to use as source to write Nvm.

4.6 Part 7 - Build

Finally buiild the whole system code and test the application.

4.7 Part 8 - Additional Notes

Before building the whole system maje sure you have added all the needed integration code (e.g. Compiler.h INLINE defines).

4.8 Part 9 - Test with ISOLAR EVE

Debug the project with ISOLAR EVE; during the execution of the debug session a binary file will be created; at the end of the test the file will be updated with the new data.

5 Contact, Support and Problem Reporting

For details of your local sales office as well as your local technical support team and product hotlines, take a look at the ETAS website:

ETAS subsidiaries	www.etas.com/en/contact.php
ETAS technical support	www.etas.com/en/hotlines.php