Dassault’s AUTOSAR Builder and EB tresos demo workflow

7 December 2021

Agenda

Enter your subtitle here

01
Project workflow

02
Requirements and architecture of the workflow demonstrator

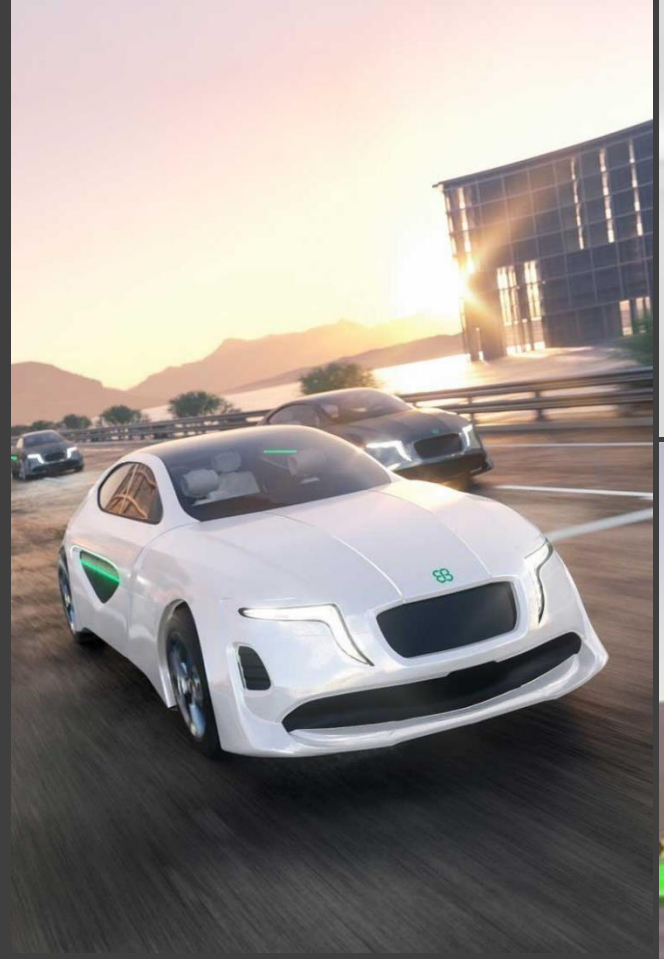
03
EB tresos workflow

04
AUTOSAR Builder (AB) workflow

05
EB tresos & AUTOSAR Builder (AB) workflow

06
EB tresos AutoCore OS & RTE

07
Generate source code and compile



Project workflow

Requirements and architecture of the workflow demonstrator

Requirements and architecture of the workflow demonstrator

Requirements

- **What shall the software do?**

1. **Output & state handling**
   - The software shall output an increasing counter if it is in state RUN.
   - The software shall reset if it is switched to state Reset.
2. **Input**
   - The software shall use keyboard inputs for controlling.
   - Only chosen characters and numbers must be used.
   - Prevent long key presses for multiple controlling.
3. **Error handling**
   - It shall be possible to detect errors during development.
4. **Reset**
   - Must be possible to do a soft reset.
   - Must be possible to do a hard reset.
5. **Storage**
   - The software must have the possibility to read ROM values.
   - The software must have the possibility to write to NvM during runtime.
   - The software must write to non-volatile-memory during soft reset (write-all).
6. **Startup**
   - The software shall use the AUTOSAR way of startup.
   - The software must read the last stored entry from non-volatile memory during startup.
   - The software components must change the state machine from Init->Run.

Tool requirements of the workflow demonstrator

Versions

Dassault AUTOSAR Builder version

- Version 2021x

EB tresos Studio version

- Version 28.1.0

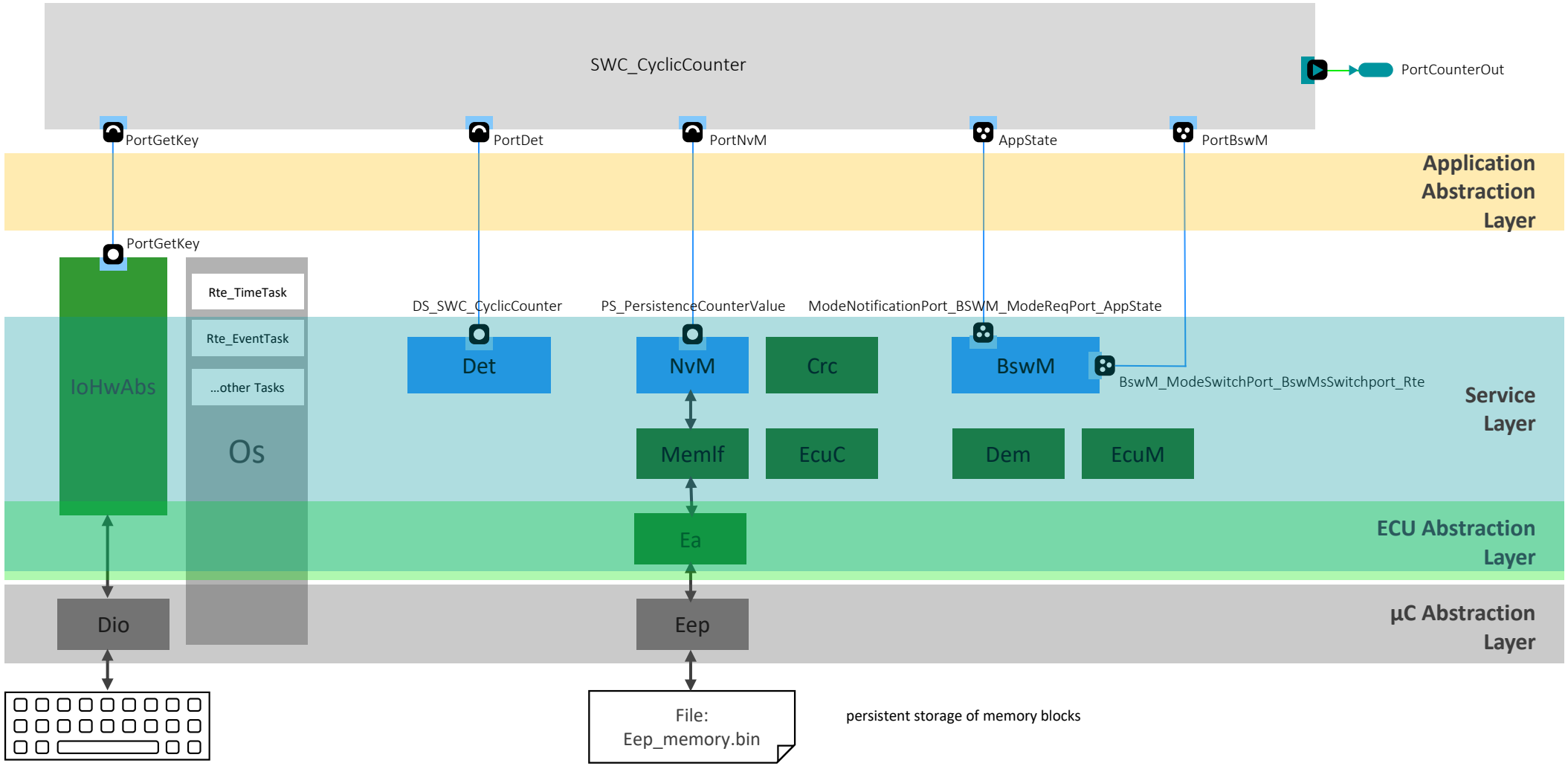
EB tresos AutoCore Generic version

- Version 8.8.3

Disclaimer

This published tool workflow corresponds to best practice and might not be the unique way to use the tools.

Project architecture



EB tresos workflow

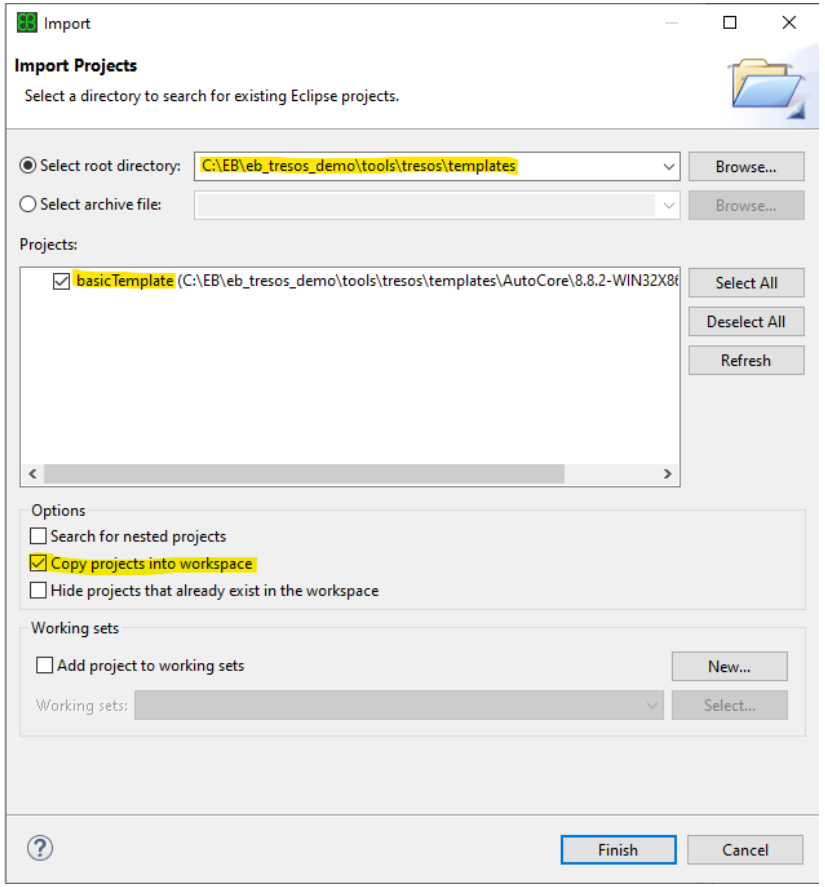
Create new EB tresos Studio project

1. **Start EB tresos Studio**
   - Run ***"Start_Tresos_new.bat"*** *and click*

2. **Create Project**
   - Open the "Workflows" tab "Window→Show view→Workflows".
   - Follow the "**Create your project**" instructions:
     1. Create new project (from Template)
     2. In the text box **"Select root directory",** browse to the project template folder ***basicTemplate***.

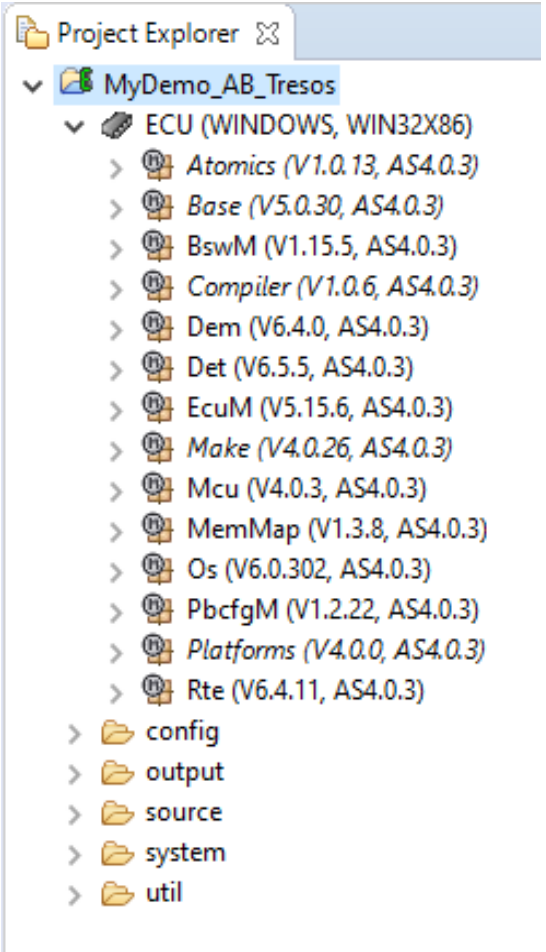
        Hint: This folder is located in the EB tresos AutoCore derivative-specific subdirectory in your EB tresos Studio installation.
     3. Select the check box "**Copy projects into workspace**" (Otherwise you will modify the template itself).
     4. Click **Finish**.

Create new project (from Template)



Rename, load & modules

1. **Rename**
   - *Mark the project in the "Project Exlorer" and press F2 (right-click → rename)*
   - *Rename your project to "**MyDemo_AB_Tresos**".*
2. **Load**
   - *Double-click the project in the "Project Explorer".*
3. **Modules**
   - **The Basic Template contains several plugins that are necessary for creating our project:**
     - *Base: Standard header file for EB treosos AutoCore*
     - *EcuM: Mode management (without OS support)*
     - *BswM: Mode management (with OS support)*
     - *Det: Development Error Tracer*
     - *MemMap: Map* code and data to specific memory sections
     - *OS: Operating System*
     - *RTE: Runtime Environment (Application → Basis Software Abstraction)*



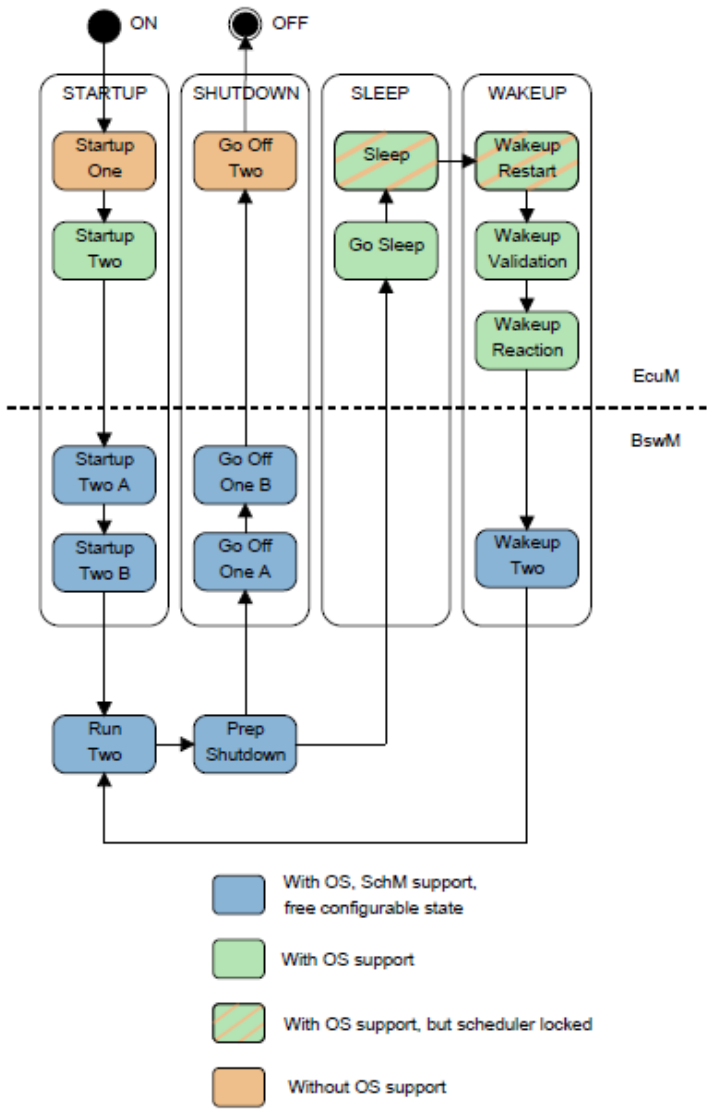
EcuM and BswM background[1]

EcuM and **BswM** are responsible for:

1. Startup behavior of an ECU
2. State management of an ECU
3. Shutdown behavior of an ECU

- EB tresos AutoCore provides the "flexible ECU management" variant of the **EcuM** module.
- This variant of **EcuM** module handles the early STARTUP phases, the late SHUTDOWN phases and the SLEEP phases.
- The whole OS scheduling is locked by the **EcuM** taking the OS resource RES_SCHEDULER.
- All other phases and states are completely configurable and handled by the **BswM** module as depicted.

- The states of the **BswM** module are completely configurable and can be adapted to system needs.



Important: On event RUN_TWO the RTE is started and the SWC can be initialized.

1 See AutoCore_Generic_Mode_Management_documentation.pdf for detailed information

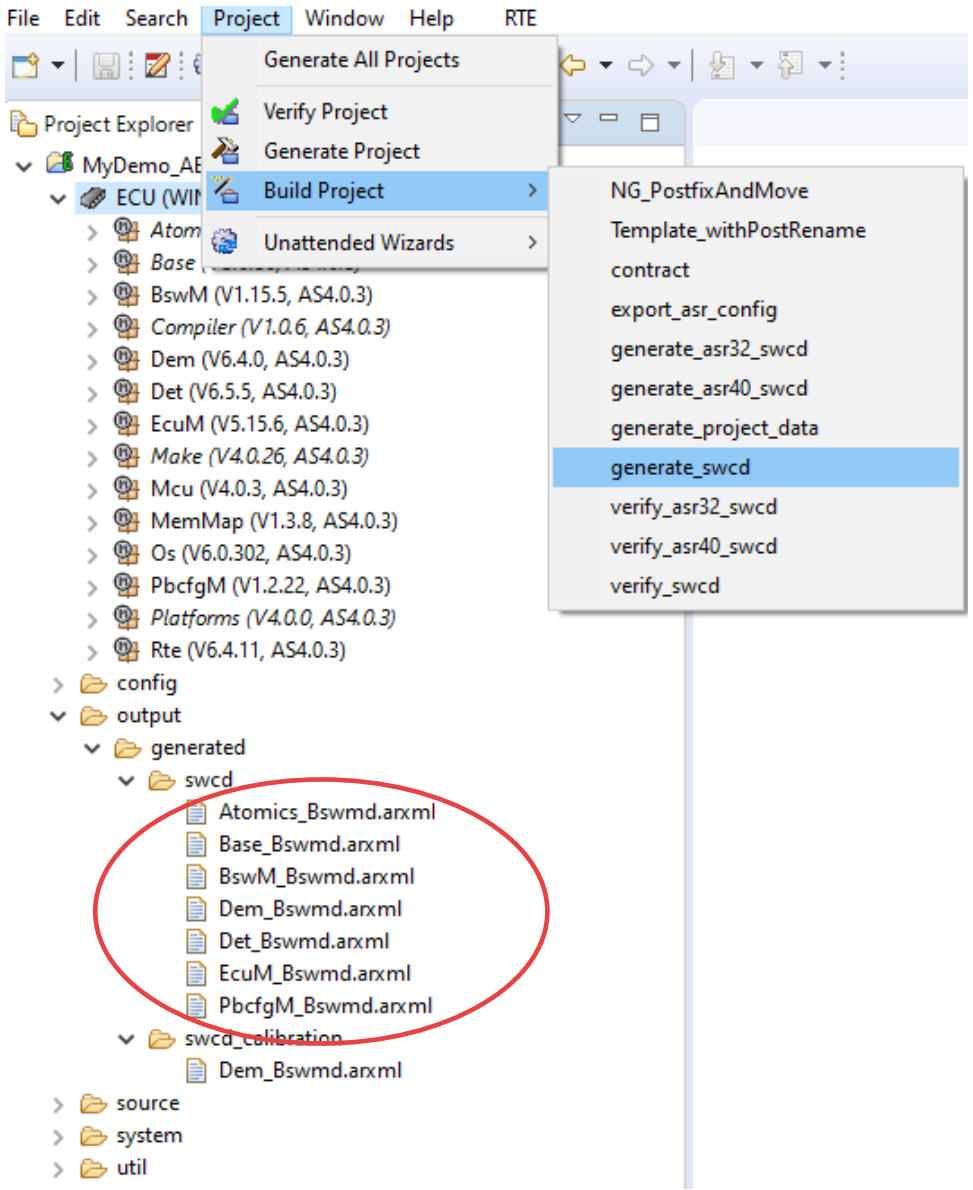
Generate BSW & SWC description

1. Generate SWCD

- *Run "Project → Build Project → generate_swcd"*
- *The **output\generated\swcd** folder contains all relevant **basic software description** (*_Bswmd.arxml) files of the configured modules*

2. Generate BSW

- *Run "Project → Generate Project"*
- *The **output\generated** folder contains all necessary files to build the project.*
  - **src/include**: *Source and header files*
  - **make**: *makefiles*
  - **orti**: ORTI stands for "OSEK Run Time Interface" and has been designed to facilitate an interface between the internal operation of an OSEK operating system and a debugger.
  - **templates**: template code for BSW and SWC software
  - **xgen**: variant handling

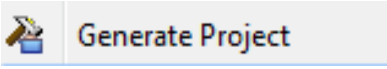


EB tresos Studio workflow

Until now you have accomplished the following:

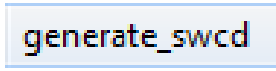
1. You created a project in EB tresos Studio which is based on a pre-configured template.

2. You generated the configuration-dependent source code for each BSW module which is part of the project.



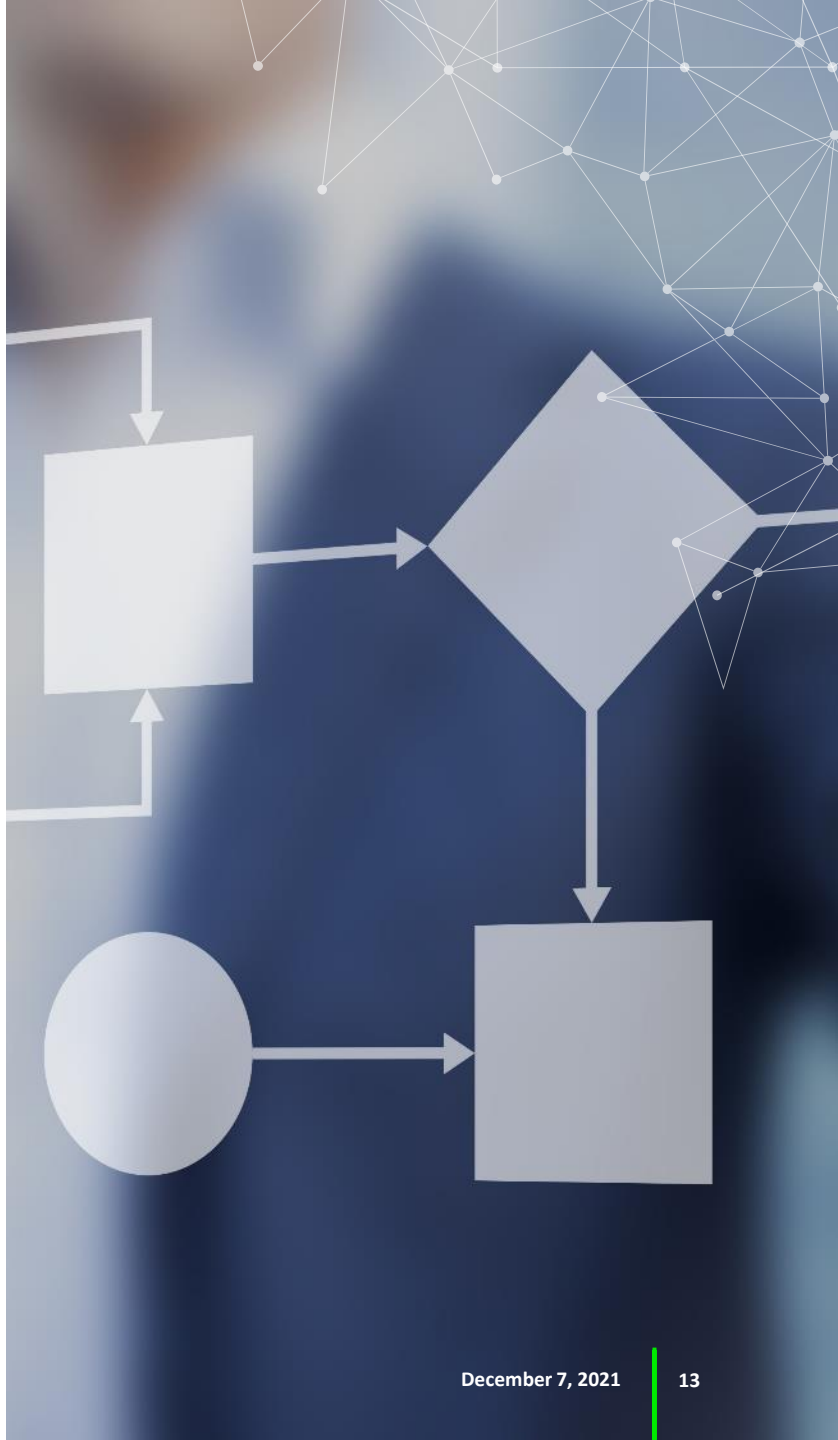
The generated files are located in the folder [project]\output\generated.

3. You generated the basic software description files (*_bswmd.arxml).



The generated files are located in the folder [project]\output\generated\swcd.

Note: The files are not required at this point of time but will become important when creating our software component with AUTOSAR Builder.



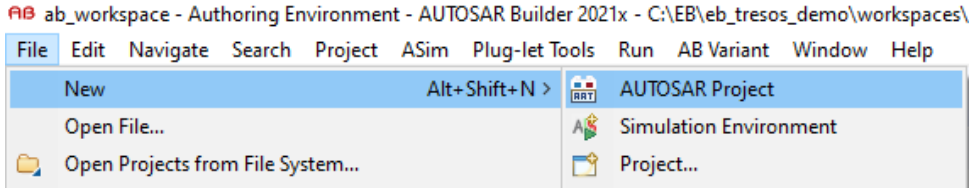
AUTOSAR Builder (AB) workflow

AUTOSAR Builder workflow

Create new AB project

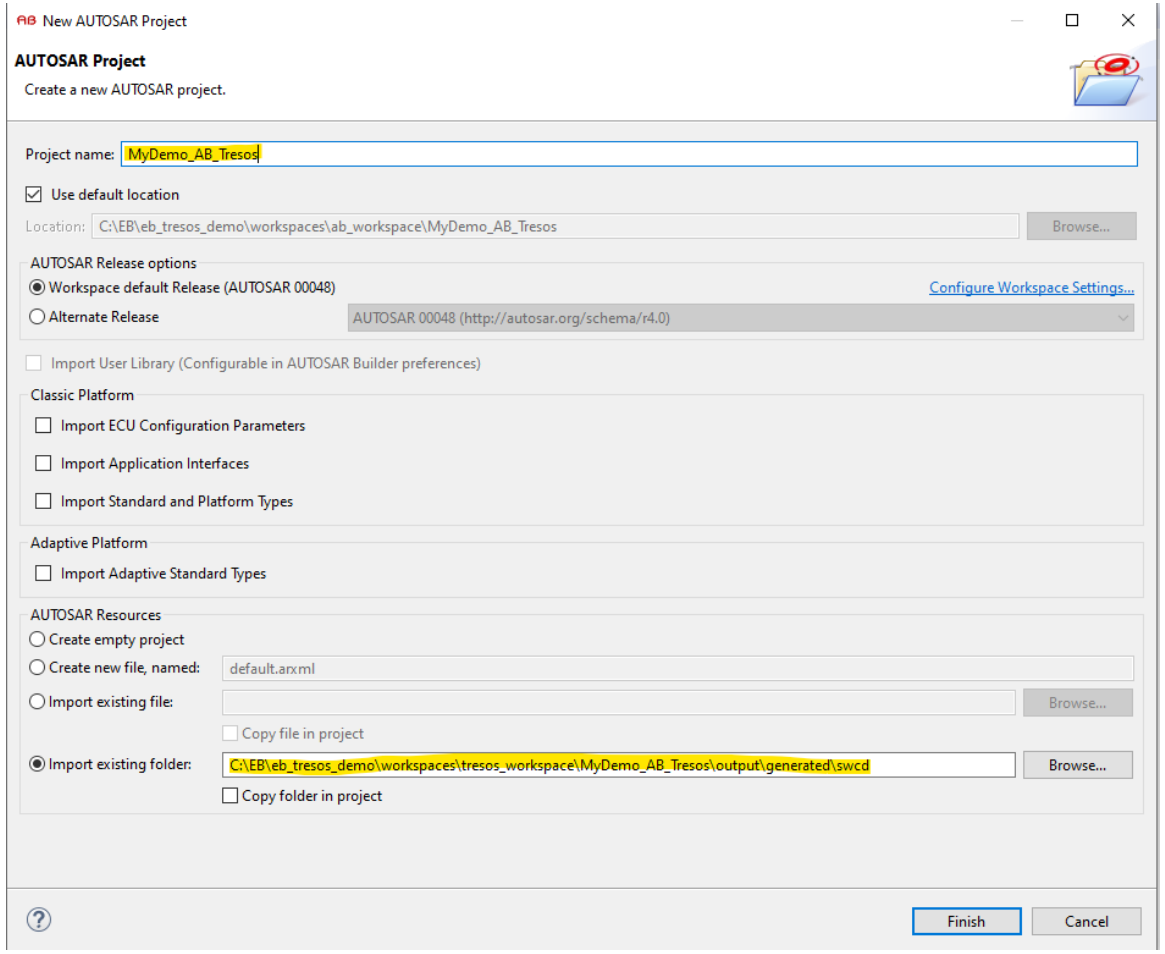
1. **Start AUTOSAR Builder**
   - Run ***"Start_AB.bat"***

2. **Create Project**
   - Click “File → New → AUTOSAR Project”

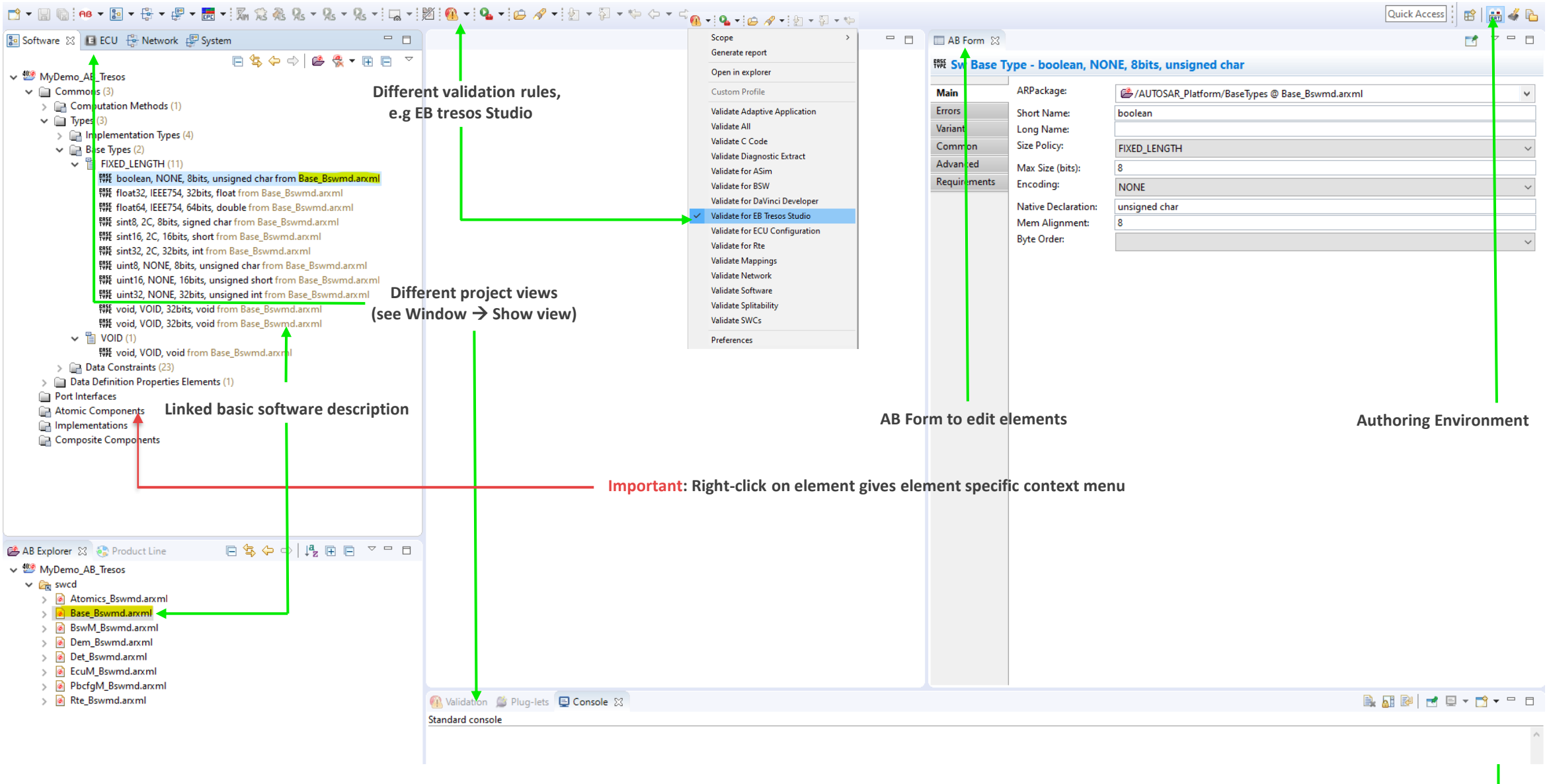


   - Enter Project name: “MyDemo_AB_Tresos”
   - Import existing **EB tresos swcd** folder (uncheck “Copy folder in project”)
     - This will **link** to the tresos workspace instead of copying the files.
   - Click **Finish**.

Create new AUTOSAR project

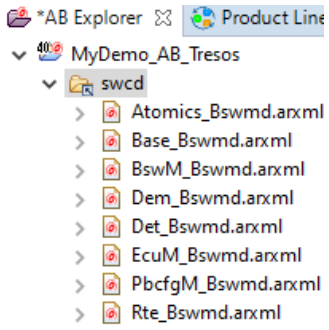


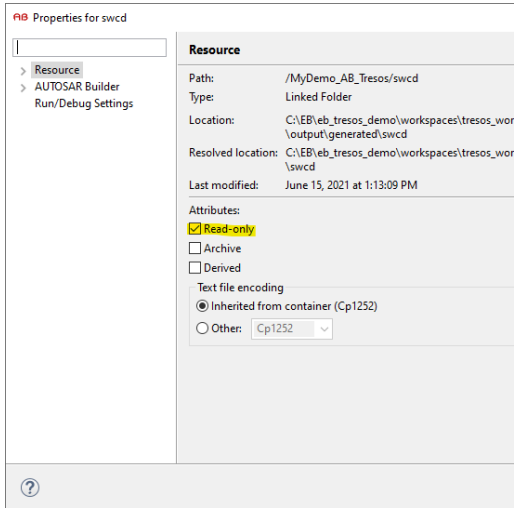
AUTOSAR Builder workflow



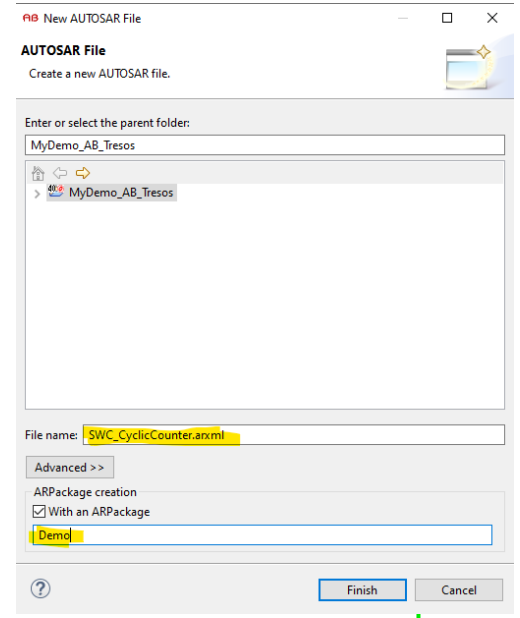
Create new SWC

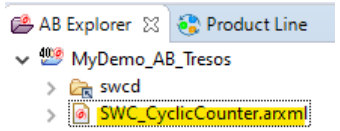
1. **Lock (read-only) EB tresos swcd folder**
   To make our life easier, we lock the swcd folder from our EB tresos workspace:
   - Right-click on AB Explorer → swcd → Properties
   - Check "Read-only"
   - Hint: Do not lock files from EB tresos swcd folder called, because these files are
     configuration-dependent.
     - XXX_**swc_interface**.arxml
     - XXX_**swc_internal**.arxml





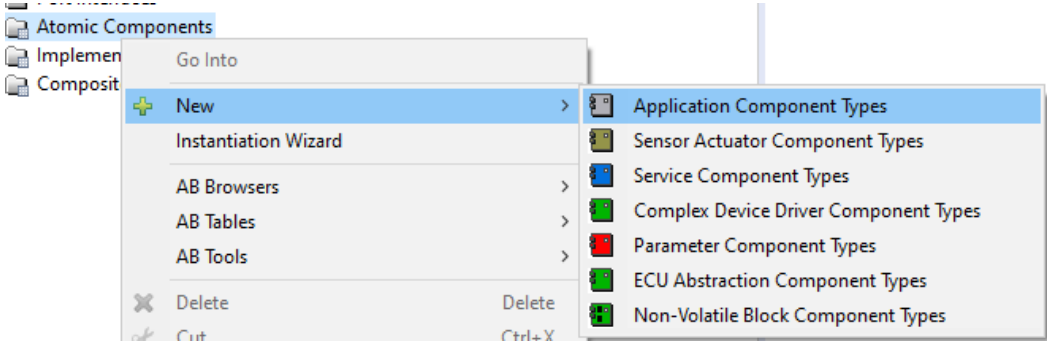
2. **Create new AUTOSAR File**
   - Right-click on Project (or AB Explorer)→ new → AUTOSAR File.
   - File name: **SWC_CyclicCounter.arxml**
   - ARPackage: **Demo**
   - The new file will be shown in the AB Explorer.
   - Due to the following steps, the new file **should** be in the parent folder **MyDemo_AB_Tresos**.

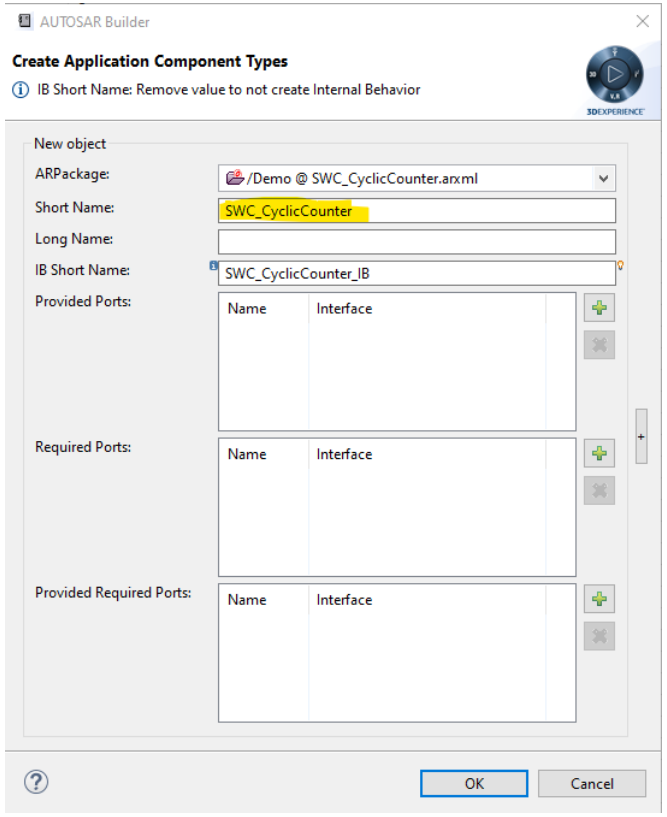


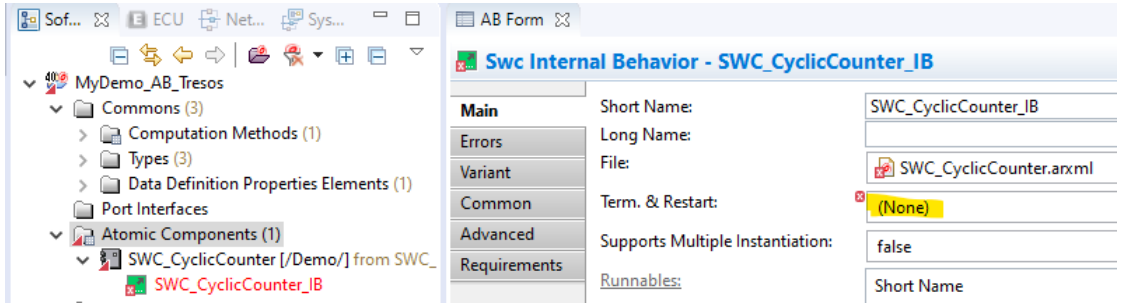


Create new SWC

3. **Create new SWC**
   - Right-click on "**Atomic Components**".
   - New → Application Component Types
   - Short Name: **SWC_CyclicCounter**
   - The ports can be added afterwards as well.

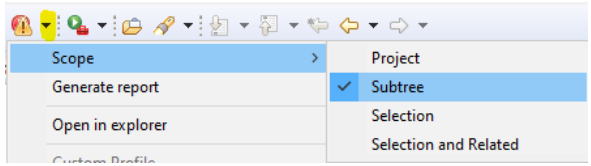




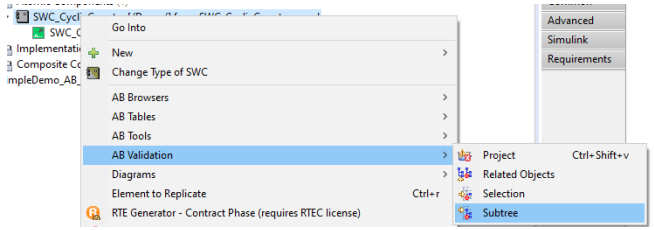


Validation example

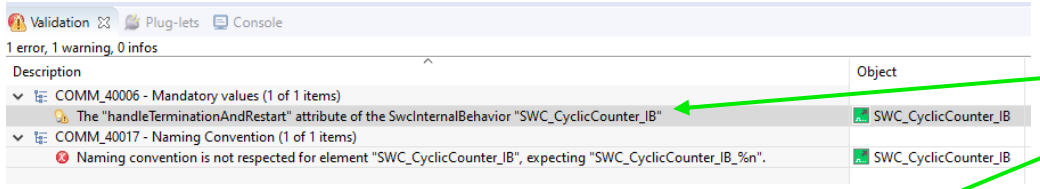
1. After creation we get some errors & warnings that we have to fix.
2. Validate Subtree, not entire project



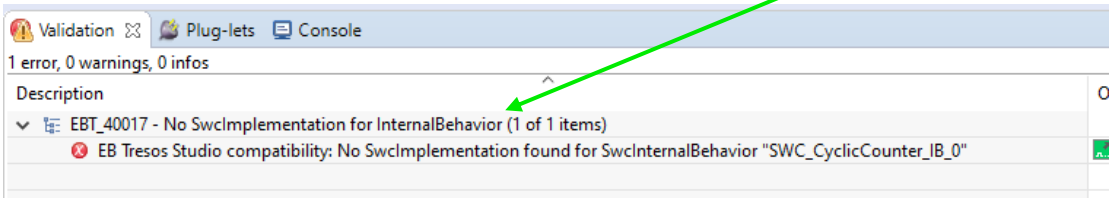
OR



3. Validation is based on the validation rules currently used: 
   - Validation SWCs



   - Validation EB tresos Studio (EBT)



Double-click guides you to the validation issue

1. Change Short name of Internal Behavior SWC_CyclicCounter_IB → SWC_CyclicCounter_IB**_0**
2. Term. & Restart: **NO-SUPPORT**

SWC implementation

1. Add new SWC implementation

a. Right-click "Implementations → new → SWC Implementation"

b. Important: Check the correct ARPackage: **SWC_CyclicCounter.arxml**

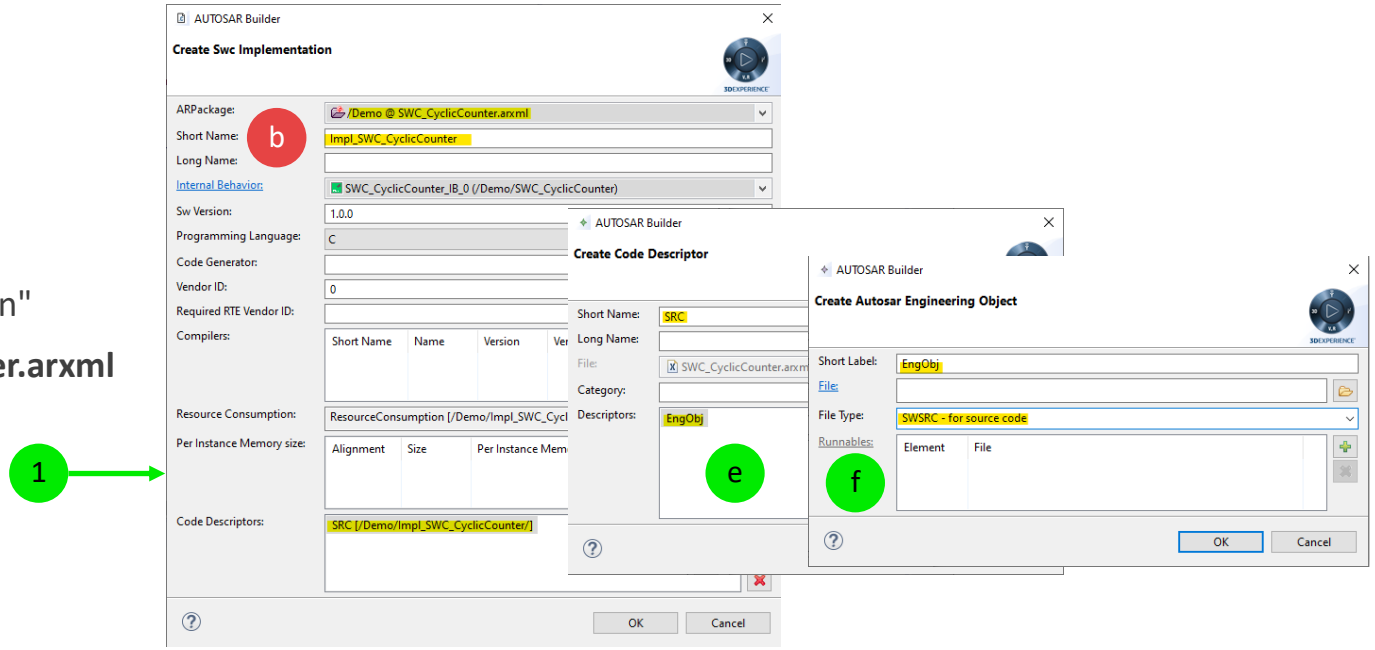
c. Add Short Name: **Impl_SWC_CyclicCounter**

d. Internal Behavior: **SWC_CyclicCounter_IB_0**

e. Add Code Descriptors: **SRC**

f. Add Engineering Object: **EngObj**

– File Type: **SWSRC**

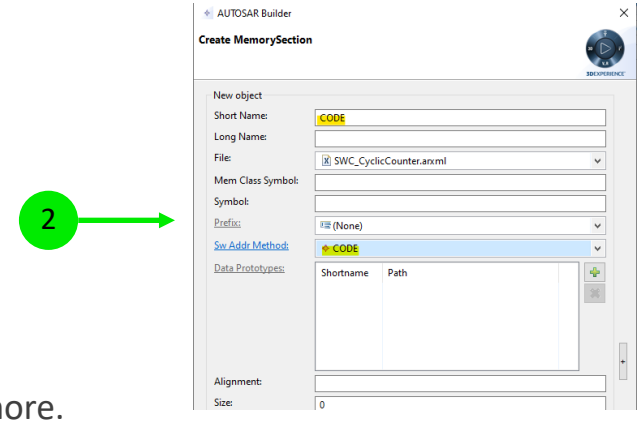


2. Add new ResourceConsumptions

a. Right-click on "ResourceConsumption → new → Memory Section"

b. Short Name: **CODE**

c. Sw Addr Methode: **CODE**



3. Validation

If you validate the subtree again, there should be no EB tresos errors anymore.

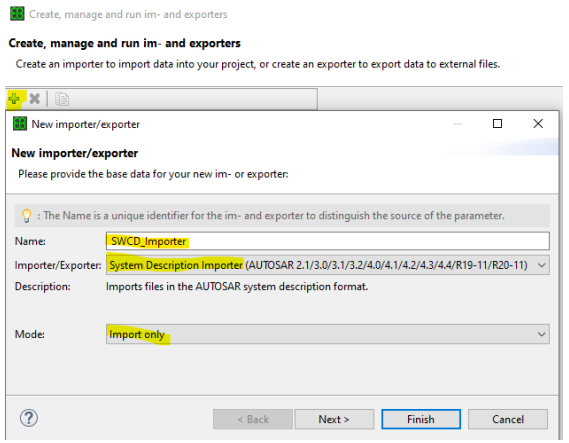
EB tresos & AUTOSAR Builder (AB) workflow

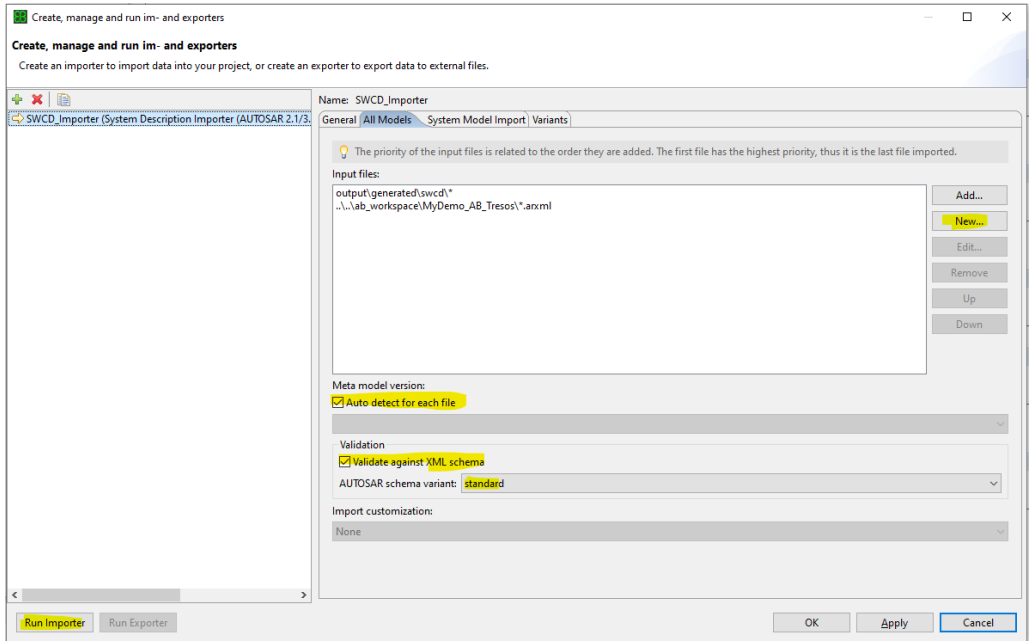
Create SWCD Importer

1. **Create new System Description Importer in EB tresos**
   - Right-click on project → "Im- and Exporters"
   - Name: **SWCD_Importer**
   - Importer/Exporter: **System Description Importer**
   - Mode**: Import Only**
   - Click **Next**

2. **Add Input Files**
   - Click **New…** and add the following paths
     - output\generated\swcd*
     - ..\..\ab_workspace\MyDemo_AB_Tresos*.arxml
   - Click **Finish**

- **Run the importer**
  - Click **Run Importer**
  - There should be no error when importing.
  - **This importer will be used for AB → EB tresos Workflow.**

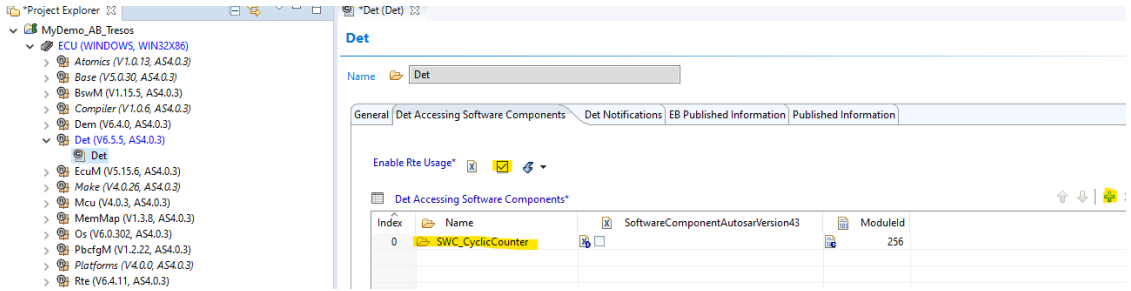




BSW ports workflow (DET example)

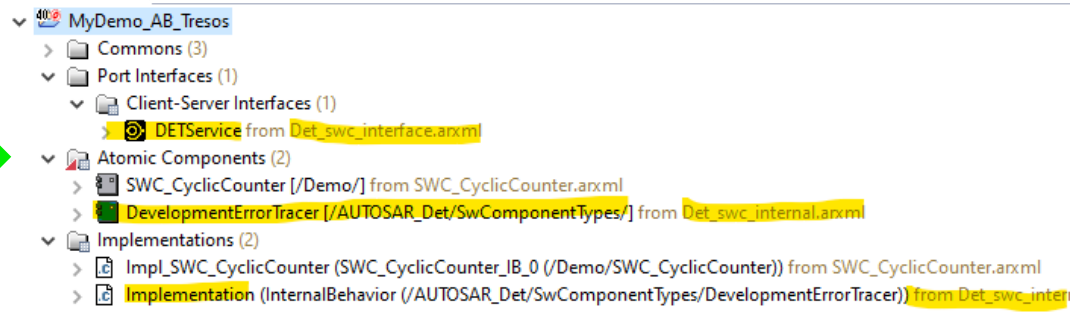
EB tresos Studio

1. **Create new BSW Port**
   We want to create a new RTE port in a BSW module that could be used by the SWC.

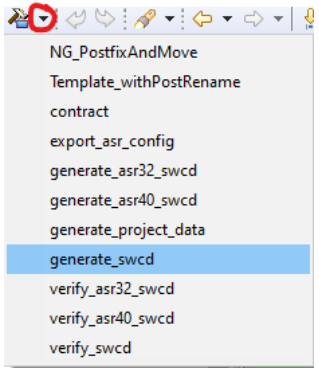


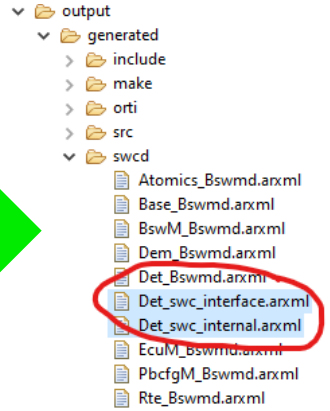
AUTOSAR Builder

1. **AB project changes on the fly**
   - New DET Port Interfaces, Atomic Component and Implementation will occur **automatically**.



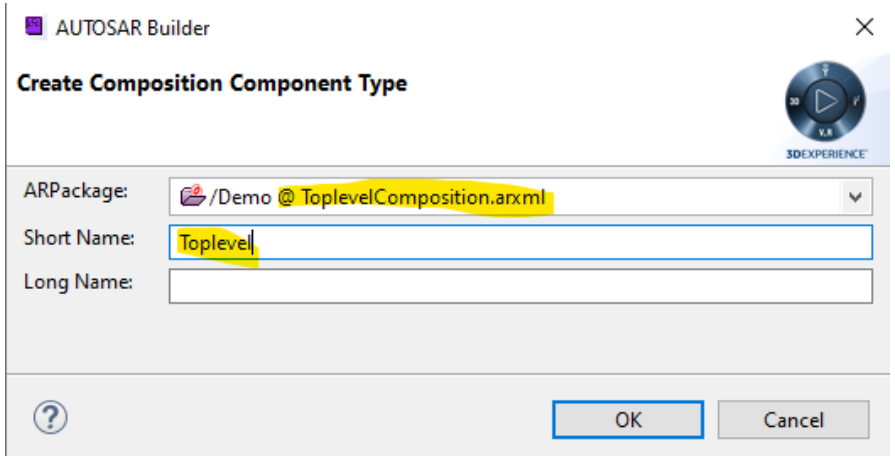
2. **Generate SWCD data**
   - Generate the SWCD data based on the new configuration.
   - Click "generate_swcd"
3. **There will be <u>two</u> additional .arxml files afterwards**
   - Det_swc_interface: Definition of the port interface
   - Det_swc_internal: Implementation, internal behavior, etc.





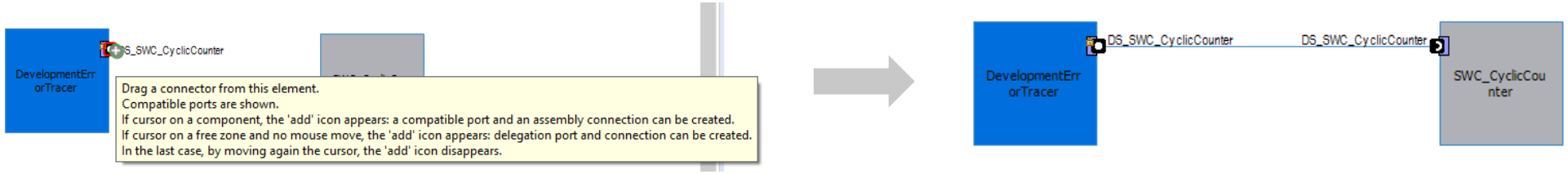
Connect ports in AUTOSAR Builder

1. **Create AUTOSAR file**
   - File → New →AUTOSAR File
   - File name: **ToplevelComposition.arxml**
   - ARPackage Name**: Demo**
2. **Create new Composition Type**
   - Composite Components → new →Composition Type
   - ARPackage: **ToplevelComposition.arxml**
   - Short Name: **Toplevel**
   - Click **OK**



Hint: Double-click on Toplevel → Opens Diagram view

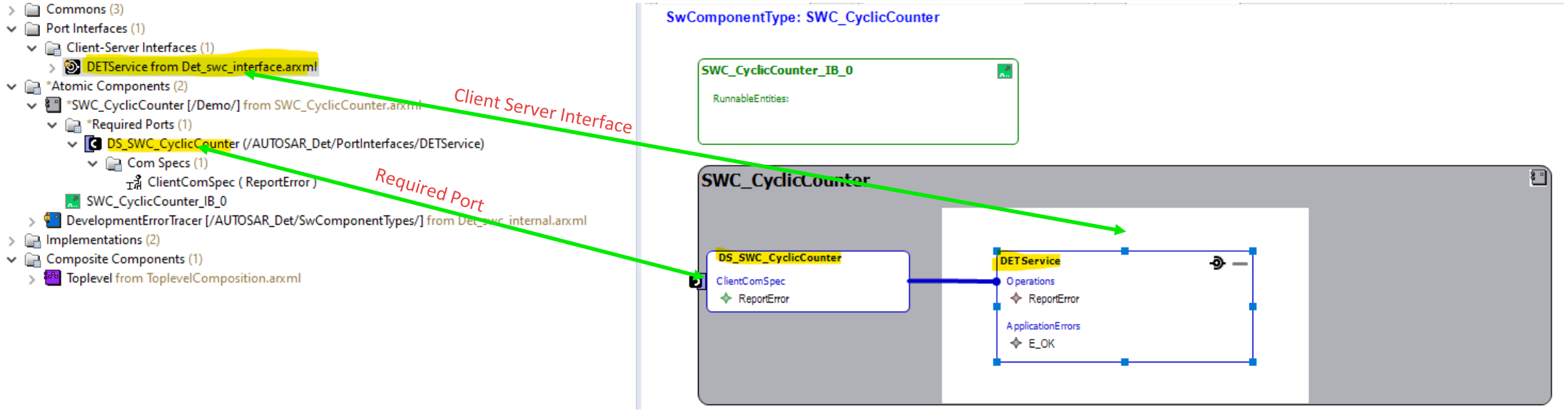
3. **Drag and drop SWC_CylicCounter** and **Development Error Tracer** from **Atomic Components** to Diagram.
4. Connect both components by dragging the connector from **Det** to **SWC_CylicCounter**.



Connect ports in AUTOSAR Builder

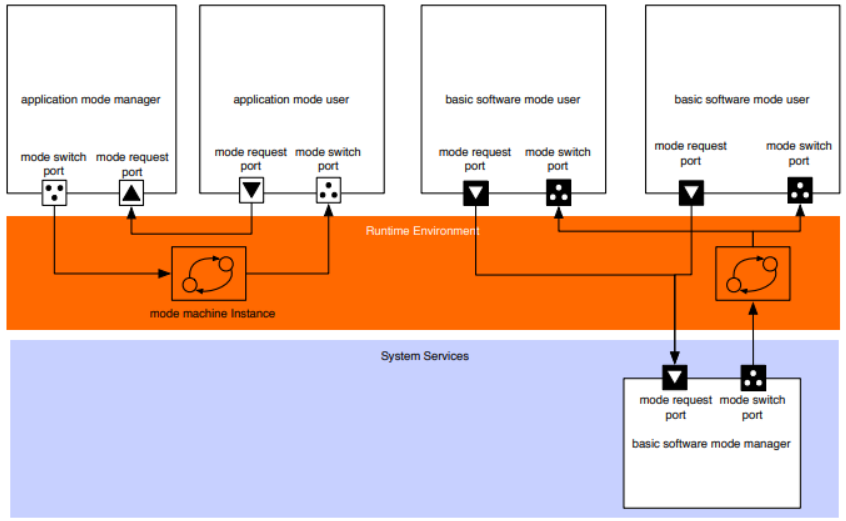
5. Visualize SWC Content

- Double-click SWC_CyclicCounter in Diagram View.
- The required ports are automatically generated for the SWC.



AUTOSAR Mode Management

In mode management there are two parties involved: **Mode managers** and **mode users**.



Mode managers:
Responsible for switching modes and are the only instances able to change the value of the global variable. A mode manager is either an SWC, which provides a **ModeRequestPort** or a Basic Software Module (e.g. BswM), which either provides also a **ModeRequestPort** in its Software Component Description or a **ModeDeclarationGroup** in its Basic Software Module Description.

Mode users:
Will be informed of **Mode switches** via well-defined mechanisms and have the possibility to read the currently active mode at any time. If a Mode user wants to change into a different mode, they can request a Mode switch from the corresponding Mode manager.

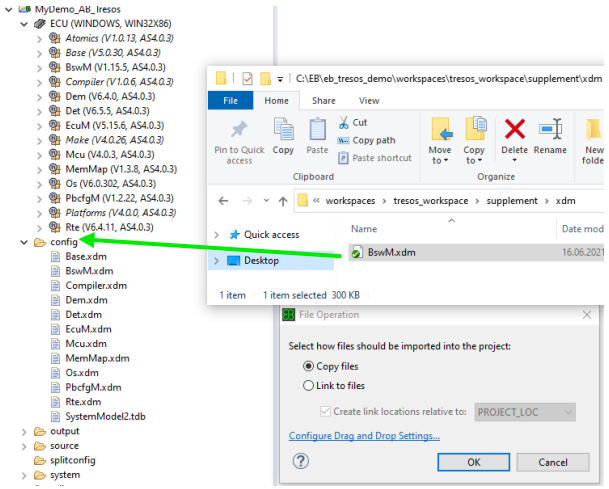
Connect BswM to SWC

1. Create BswM RTE port

Because we want call our runnables based on modes defined by the BswM, we need to create a port to inform the SWC about mode changes.

a. Copy (drag & drop) the **BswM.xdm** from **supplement/xdm** folder to your EB tresos project config folder and overwrite the current one.

- The .xdm file contains the necessary configuration for creating BswM ports.



2. Compare BswM config

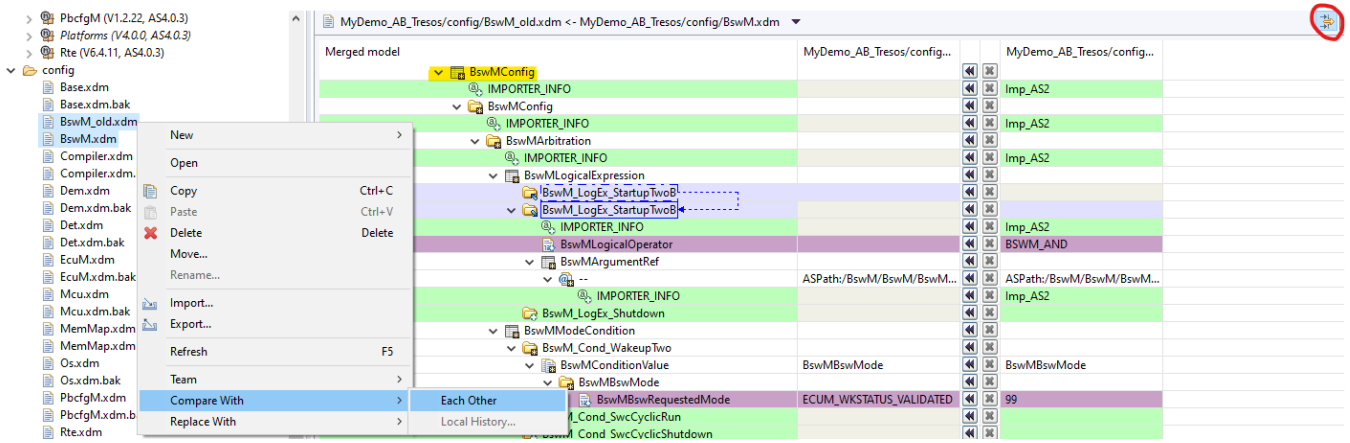
If you are interested in the config changes within the copied BswM, you can use the EB tresos Studio built-in compare tool

a. Rename the BswM.xdm.bak → Bswm_old.xdm.

b. Mark both BswM files (Press Ctrl button).

c. Press button "Only show unprocessed changes".

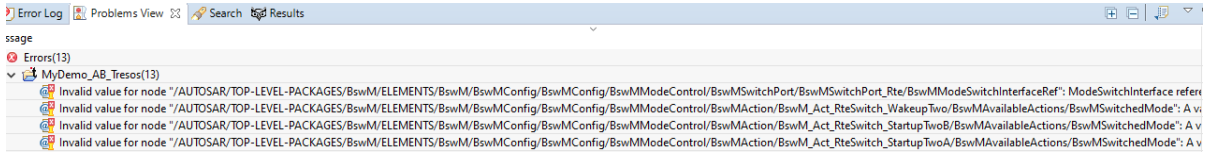
d. The changes tree will show all the additional configurations.



Connect BswM to SWC

3. Create Mode Declarations Groups

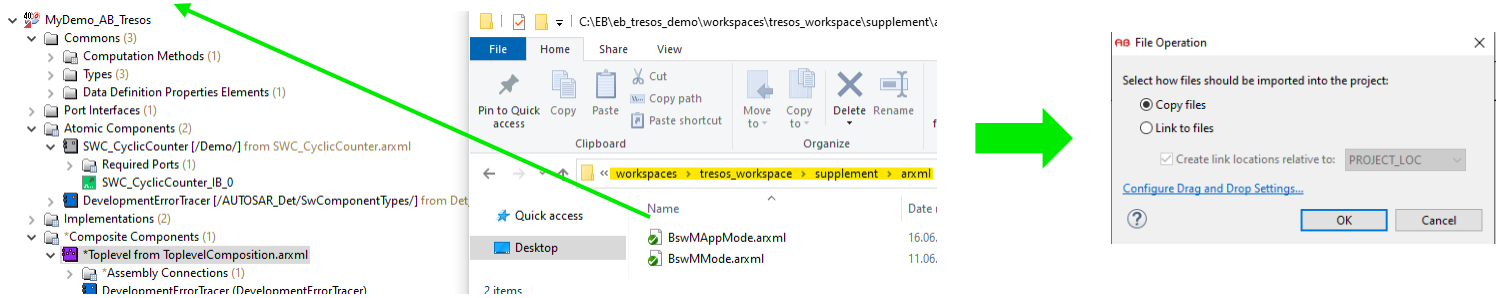
After we copy the BswM.xdm, we will see some EB tresos errors in the "**Problems View**".



This is because the BswM has no idea of the **Modes and Port Interfaces.** We need to give that information to the BswM.

- **Copy** (Drag & Drop) the content of the **supplement/arxml** folder to your AUTOSAR Builder project (prepared **ModeDeclarationGroups**)

  Note: We copy that directly there because we import everything from the AUTOSAR Builder project in our EB tresos project and we need these groups for further processing in AUTOSAR Builder.



- **BswMMode.arxml**: Contains the modes defined by BswM
- **BswMAppMode.arxml**: Contains the modes of the application SWC_CyclicCounter.

Connect BswM to SWC

4. Inspect the Mode Declaration in AB

- The **Mode Declarations** can be viewed in AB.
- The modes correspond to the ones defined in EB tresos BswM configuration.

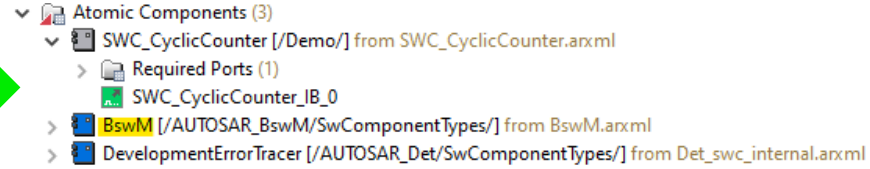
5. Import ModeDeclarationGroups & Port Interfaces in EB tresos

- Run **"SWCD_Importer"** (Right-click on project → Im- and Exporters..)

  Note: If you see import warnings, you can ignore them as they are due to a missing mapping.
- The BswM errors in the "Problems View" should disappear.
- Now the BswM has all the information he needs to generate its SWCD.

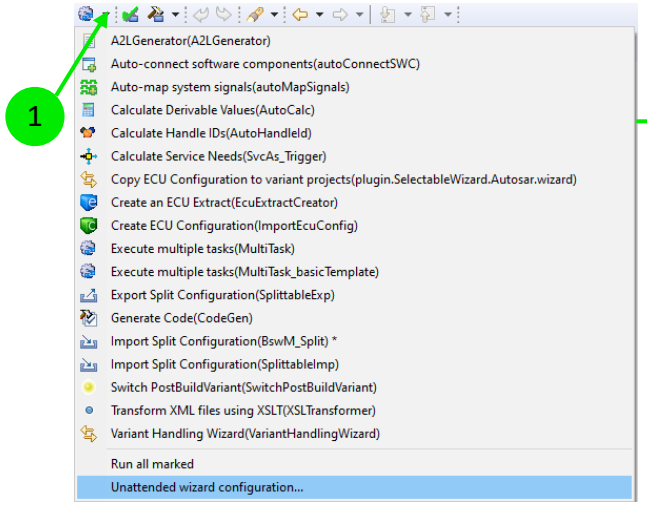
6. Generate SWCD in EB tresos

- Run "generate_swcd".
- In your AB project a new "Atomic Component" (BswM) will appear.
- Now we can use the BswM, for example, to start our runnables or to inform the BswM about mode changes (e.g. shutdown).

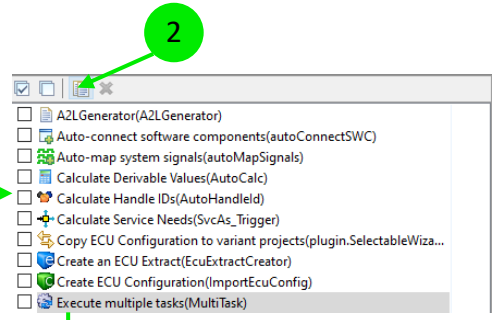


Create EB tresos MultiTask wizard

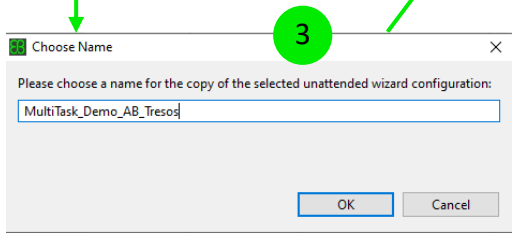
As we have to do several steps such as **importing**, **generate_swcd**, etc., we create an MultiTask wizard in EB tresos that can automatically execute all manually steps sequentially. We will update this wizard with an additional task.



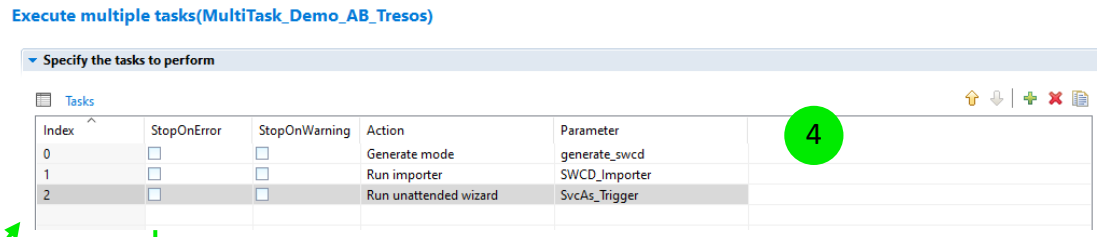
1. Open "**Unattended wizard configuration**"



2. Copy "**MultiTask**" wizard

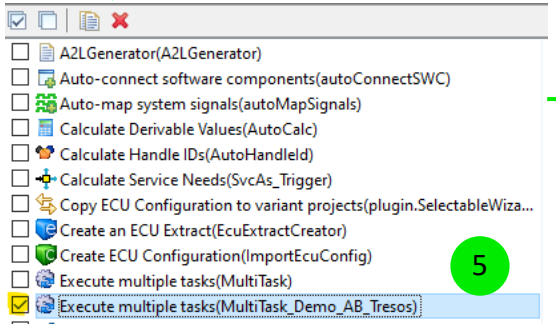


3. Choose appropriate name

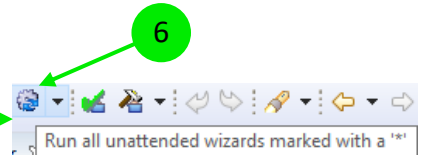


4. Configure the wizard tasks

0: Generates SWCD
1: Run our SWCD_Importer
2: Calculate Service Needs for OS (Events, Alarms, etc.), Dem, EcuM



5. Enable wizard



6. Run **all** enabled wizards

Connect BswM to SWC

7. Create SWC ModeSwitchPort (MSP)

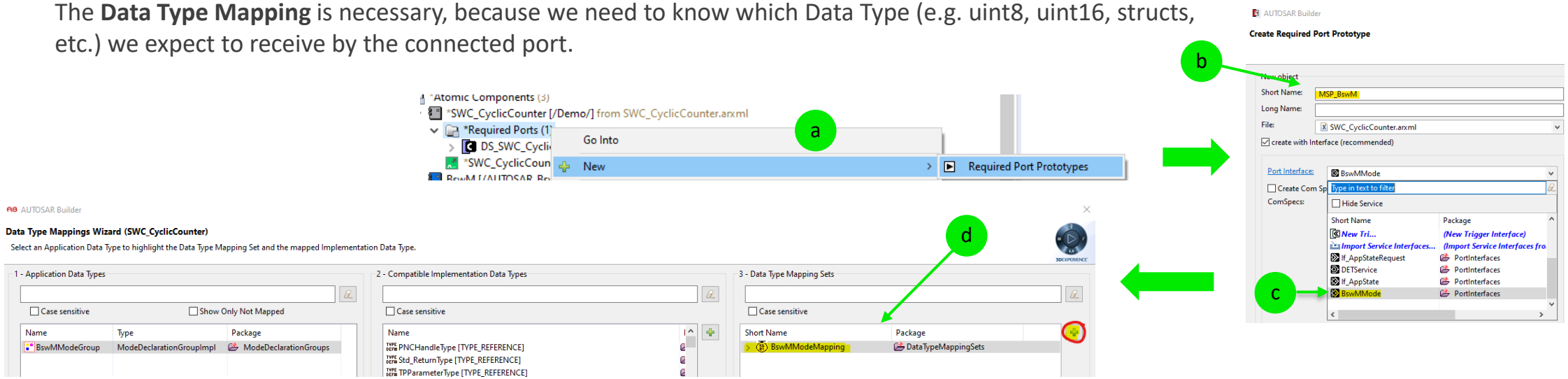
First, we need to create a required port (R-Port) that BswM can inform our SWC about mode changes. We can either do that manually or by connecting the BswM MSP to our SWC (see "BSW Port Workflow Example"). We want to do that manually to see what we can do if we want to create new R- or P-Ports.

a. Right-click on "Required Ports" and add a new Port Prototype.

b. Short Name: **MSP_BswM**

c. Port Interface: Choose **BswMMode** Port Interface

d. Add the **Data Type Mapping**
The **Data Type Mapping** is necessary, because we need to know which Data Type (e.g. uint8, uint16, structs, etc.) we expect to receive by the connected port.



Connect BswM to SWC

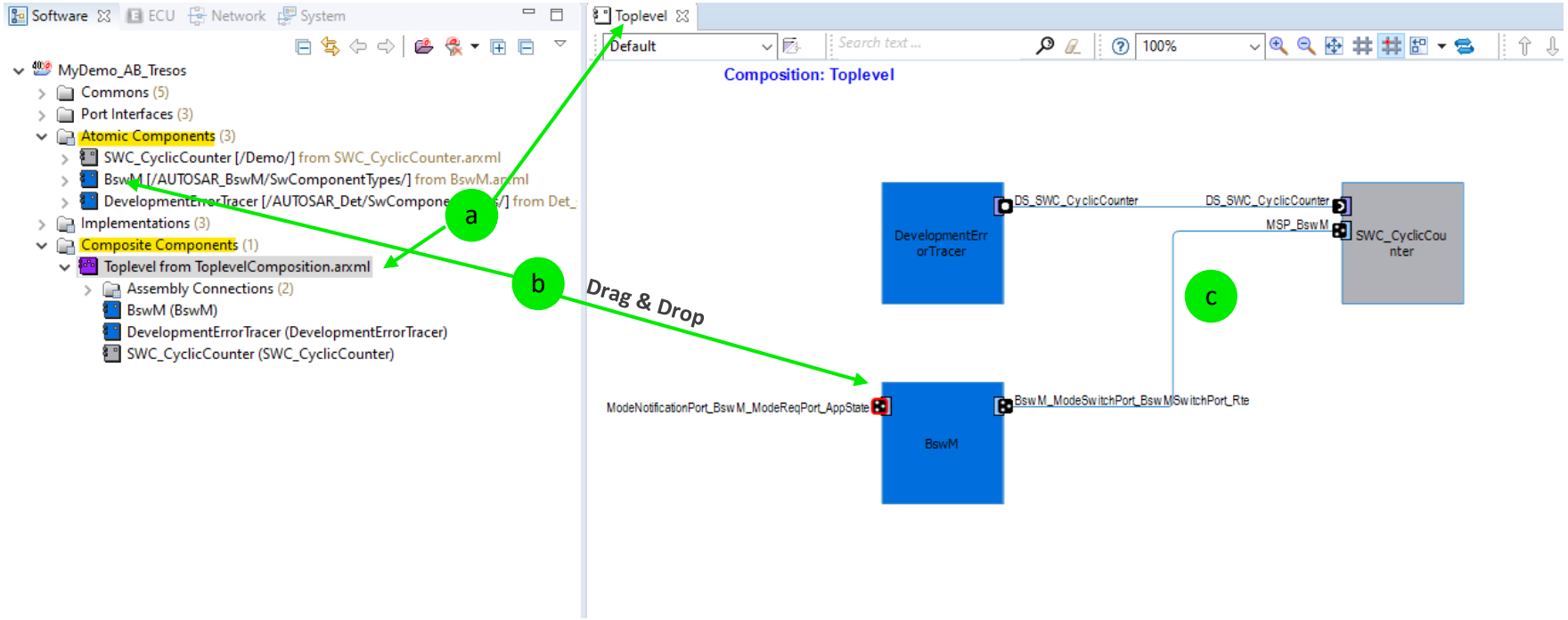
8. Connect SWC ModeSwitchPort (MSP)

Now, we want to connect the MSP of the BswM with the already created MSP_BswM of our SWC.

a. Open **Top-level Diagram View** from Composite Components (double click).

b. Drag & Drop BswM from Atomic Components to top-level view.

c. Connect BswM MSP with MSP_BswM of SWC.



Create runnables

Now we want to add some **runnables** to our SWC. As you know, runnables describe functions that can be activated by an RTE event.

In the following we will deal with the following topics:

- We will add an (**explicit**) **interrunnable variable** for **"CurrentCounterValue":**
  - **Explicit** means the runnable entity can directly access an interrunnable variable. Changes are immediately visible to other runnable entities with explicit access to the interrunnable variable.
  - Communication across all runnables within an SWC is possible.

- We will add the following **runnables**:
  - **Event-triggered** ("SWC_CyclicCounter_Init"),
  - **Time-triggered** ("SWC_CyclicCounter_Cyclic")

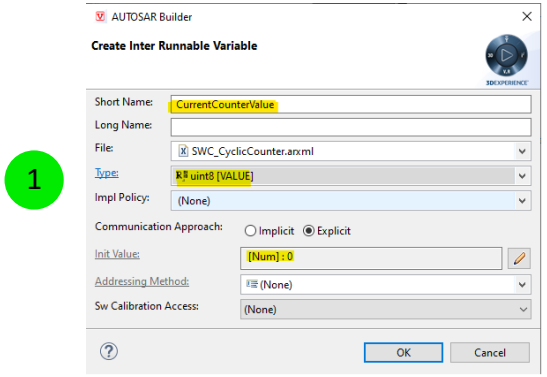
- We will add **access points:**
  - **Server-Call Points Access**: Report Error via Det Port
  - **InterRunnable Variable Access**: Read and Write access on variable

Create runnables

1. Create Inter Runnable Variable (IRV)

I. Open the context menu of Internal Behavior **SWC_CyclicCounter_IB_0**

II. Add New → **Inter Runnable Variable**



2. Create Runnable SWC_CyclicCounter_Init

I. Open the context menu of Internal Behavior **SWC_CyclicCounter_IB_0**

II. Add New → **Runnable Entities**

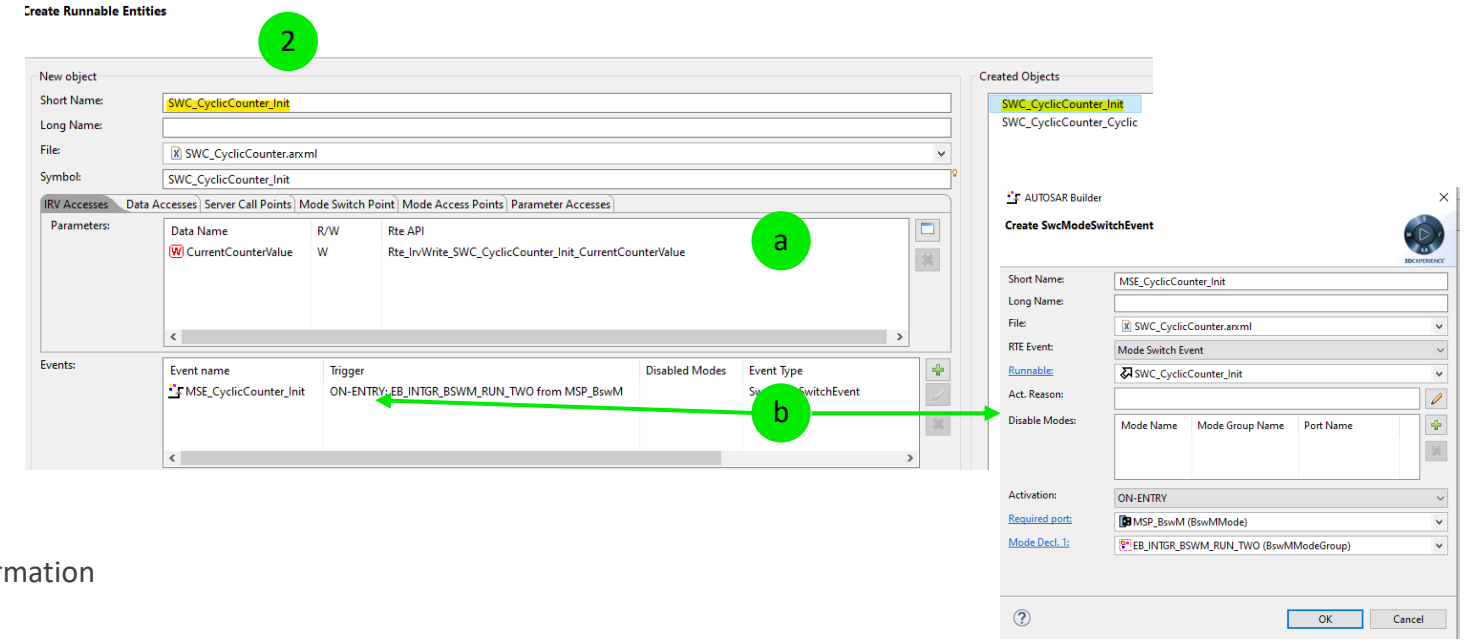
III. Tab **IRV Accesses:**

a. Parameters:

- Write Access on IRV

b. Events:

- **Event Name:** MSE_CyclicCounter_Init
- **Event Type:** Mode Switch Event
- **Activation:** ON-ENTRY
- **Required port:** MSP_BswM
- **Mode Decl.:** RUN_TWO
  - See EcuM & BswM background slide for information



Create runnables

3. Create runnable SWC_CyclicCounter_Cyclic

I. Open the context menu of Internal Behavior SWC_CyclicCounter_IB_0

II. Add New → **Runnable Entities**

III. Tab **IRV Accesses:**

a. Parameters:

– Read/write access on IRV

IV. Tab **Server Call Points (SCP)**:

b. Accesses:

– **Synchronous** Server Call Point for operation Det **ReportError**

c. Events:

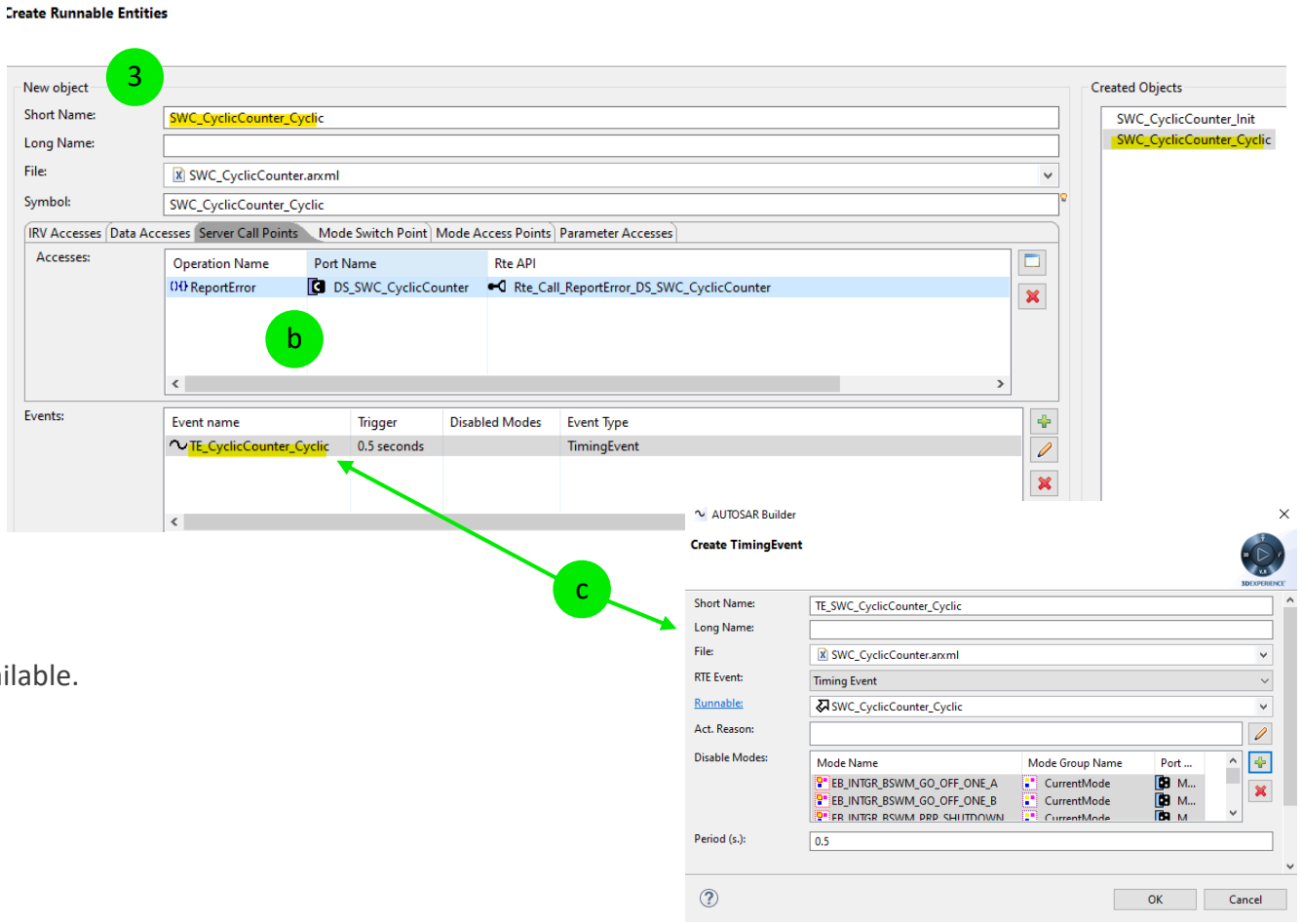
– **Event Name:** TE_CyclicCounter_Cylic

– **Event Type:** Timing Event

– **Disable Modes**: Add all modes except RUN_TWO

- The runnable will not triggered if one of the other modes will be available.

– **Period:** 0.5 (seconds)



AUTOSAR Builder

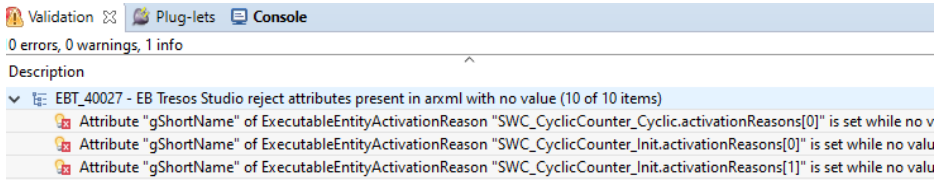
Validate SWC

1. AUTOSAR Builder validation rule for EB tresos Studio

a. Open the validator and choose "**Validate for EB tresos Studio**".

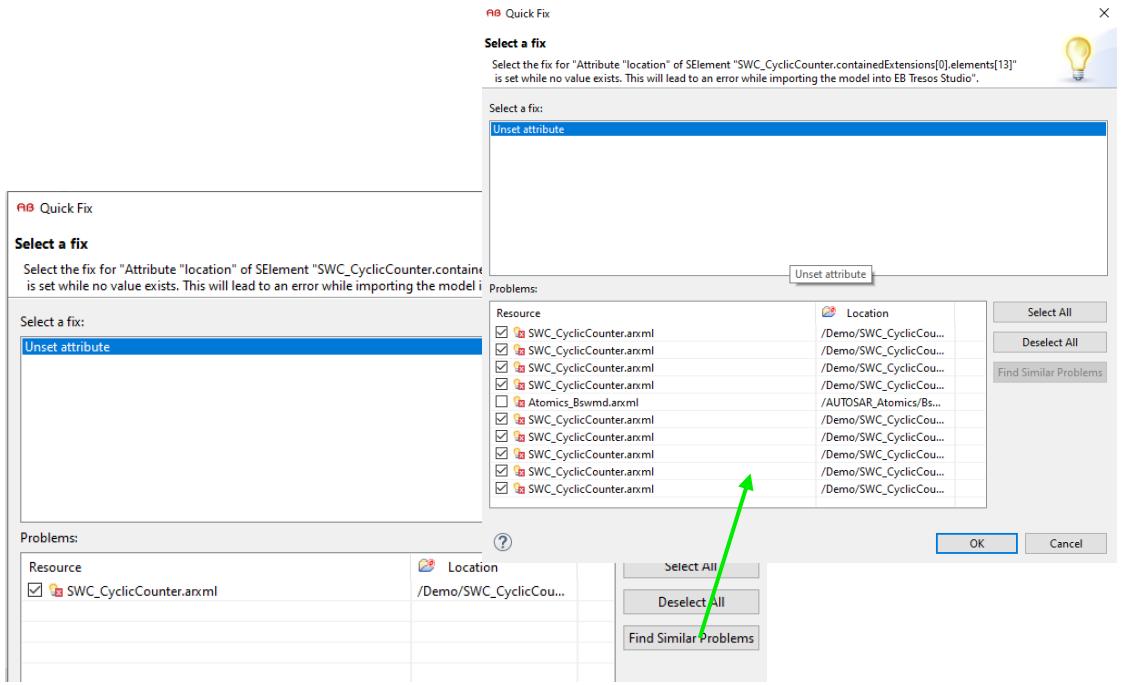
b. Right-click on your project → AB Validation → Project (or click the validator button after you marked your project).

c. Some validation errors will be shown in the "**Validation**" view.



d. Double-click on the error will guide you to the error.

e. Right-click →**Quick Fix (Crtl +1)** will give you some hints how to fix the issue.



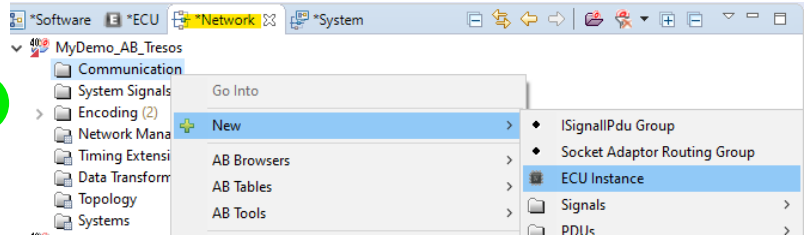
You can use "**Find Similar Problems**" to fix all problems at once.

SWC to ECU mapping

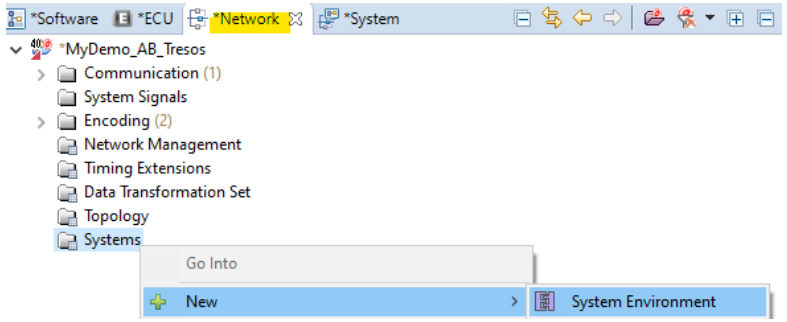
After we created our SWC and connected it to the BSW prototypes, we need to map the SWC prototypes to a specific ECU.

1. Create new AUTOSAR file
   a. Filename**: System.arxml**
   b. ARPackage**: Demo**
2. Create new ECU instance
   a. Open **"Network" view**
   b. ARPackage: **Demo@System.arxml**
   c. Name**: DemoEcu**
3. Create new System Environment
   a. ARPackage: **Demo@System.arxml**
   b. Name: **DemoSystem**
   c. Category: **SYSTEM_DESCRIPTION**
   d. RootSwComposition Prototype: **Toplevel**
   e. System Version: **1.0.0**
   f. Fibex Element: **Choose DemoEcu Instance**

2

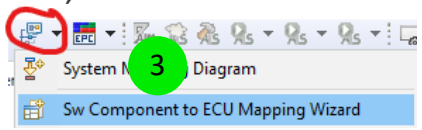


3

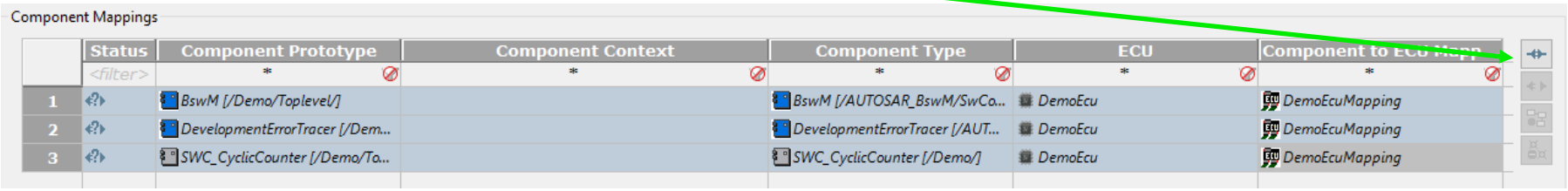


SWC to ECU mapping

Now all SWC can be mapped to the ECU instance. Therefore, we can use the integrated AUTOSAR Builder wizard.



3. Open SW **"Component to ECU Mapping Wizard"**
   a. Double-click on **DemoSystem**.
   b. Mark **all** SWC Component Prototypes and connect them to the ECU instance.



4. The **DemoEcu** now contains all SWC prototypes we need to configure the RTE.
5. Next, we are going to import the entire system into our EB tresos Studio project.

EB tresos AutoCore OS & RTE

Create OS tasks

As we know that every runnable needs to be mapped to a task, we have to create two additional OS tasks in our EB tresos AutoCore OS configuration. The two tasks are needed to map the cyclic and the event-driven runnable that we have created.

Create two additional OS tasks

- Open the OS config in EB tresos Studio "Project Explorer" and go to tab "OsTask".
- Add **two** additional tasks.

General | EB Published Information | OsAlarm | OsAppMode | OsApplication | OsCounter | OsEvent | OsSpinlock | OsIoc | OsIsr | OsResource | OsTask | OsScheduleTable

OsTask*

Index	Name	OsTaskActivation	OsTaskPriority	OsStacksize	OsTaskSchedule
0	Init_Task	1	127	1024	NON
1	SchMDiagStateTask_20ms	1	50	512	FULL
2	Rte_Event_Task	1	51	1	FULL
3	Rte_Time_Task	1	52	1	FULL

1. **Rte_Event_Task**:
   - Os TaskPriority: 51
2. **Rte_Time_Task**:
   - Os TaskPriority: 52

In this example all other params could keep their default values. Any needed params such as **OsStackzize** will be calculated automatically.

Update MultiTask wizard

We need to update/extend our previously created MultiTask wizard

Specify the tasks to perform

Tasks

Index	StopOnError	StopOnWarning	Action	Parameter
0	☑	☐	Generate mode	generate_swcd
1	☑	☐	Run importer	SWCD_Importer
2	☑	☐	Run unattended wizard	EcuExtractCreator
3	☑	☐	Run unattended wizard	AutoHandleId
4	☑	☐	Run unattended wizard	SvcAs_Trigger

Add **EcuExtractCreator** and **AutoHandleID**

- **EcuExtractCreator:** Creates an ECU extract that is needed to select a system and ECU for RTE configuration.
- **AutoHandleID:** Calculates the IDs for the different ACG plugins used such as CanIf, Dem, etc. (not necessary here, only for completeness).

Important: Please keep the task sequence as shown above, otherwise it is not guaranteed to have a correct workflow.

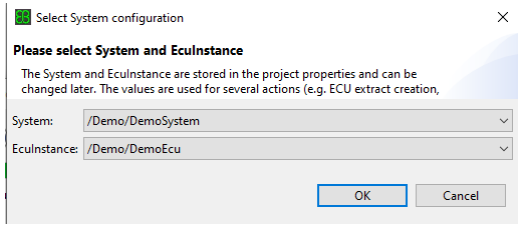
Import System into RTE

1. Run the MultiTask wizard

– All necessary files will be generated and imported.

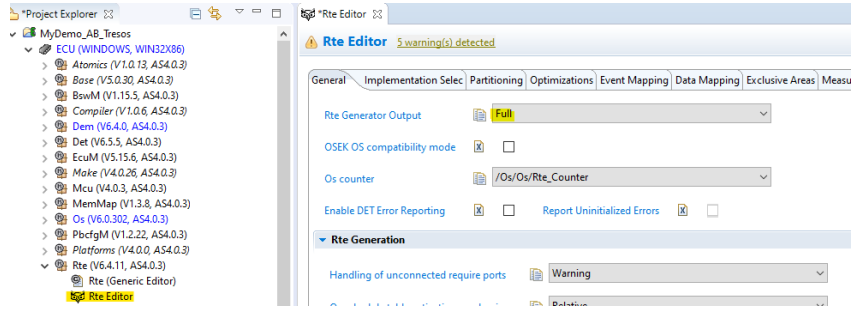
2. Select System and EcuInstance

– You need to select a System and EcuInstance for configuring the RTE.



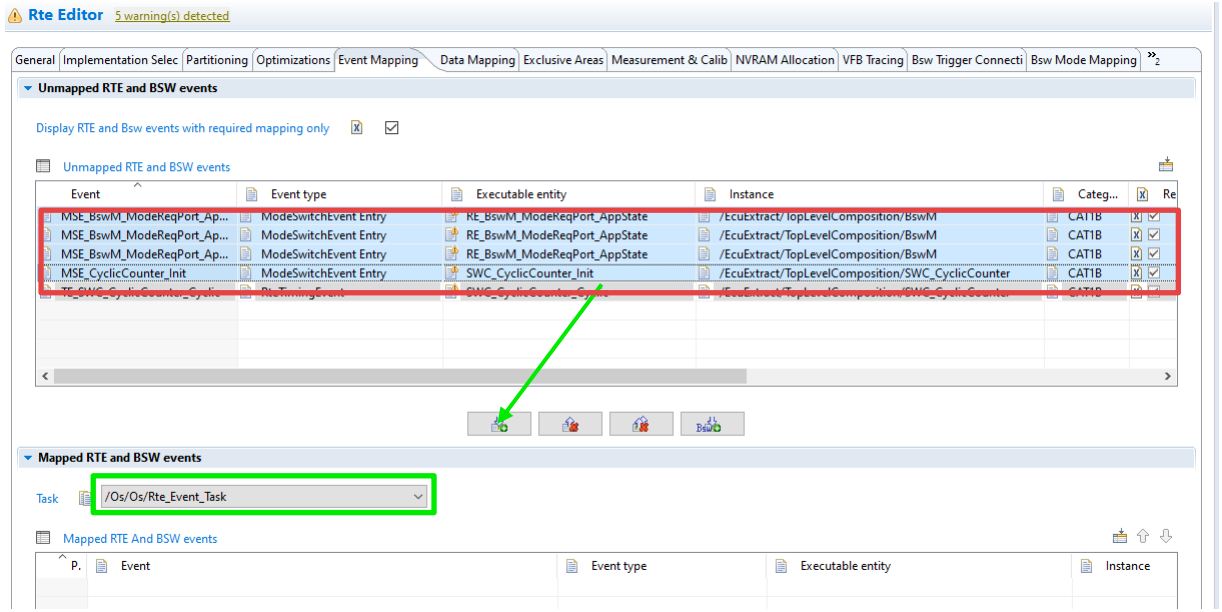
3. Open RTE Editor

– Change "**Rte Generation Output**" to **Full**



4. Runnable to task mapping

– Open tab "**Event Mapping**"



– Map all ModeSwitchEvents (MSE) to **Rte_Event_Task** (**Note**: AppState events are not necessary here, but they are part of the entire demo)

– Map RteTimingEvent (TE) to **Rte_Time_Task**

– **Close** RTE, save your project, and run the "**Calculate Service Needs**" wizard.

EB tresos & AUTOSAR Builder workflow

Until now you have accomplished the following:

1. You created a new project in AB and know how to include BSW SWCD.
2. You created you own SWC in AB and learned how to visualize them.
3. You learned how to validate your SWC against EB tresos and other validation rules.
4. You created your first connection between your SWC and the ports provided by the DET BSWMD.
5. You got some insides about Mode managers and Mode users that are necessary for switching modes in AUTOSAR.
6. You got familiar how to add and access necessary SWC subelements, such as:
   - Creating Runnables
   - Creating ModeSwitchPorts (MSP)
   - Creating Inter Runnable Variables (IRV)
   - Creating Server Call Points (SCP)

11. You learned how to connect the Basis Software Manager (BswM) to your SWC and how to use ModeDeclarations.

12. You learned how to map your SWC to an ECU creating a system environment.

13. You got some insights about "How to create EB tresos unattended wizards" and how to use them to import your system into the RTE.

14. You got familiar with the process of:
    - Creating OS tasks for "runnable to task mapping"
    - Configuring the RTE and map the runnable events to the pre-configured tasks

Summary: You learned the basic workflow concept between EB tresos Studio and AUTOSAR Builder, and how comfortable it is to build up your own system really fast by connecting BSW with application SWC in an easy and visualized manner.

Iterate through the workflows

Working agile or by increments

Normally, an ECU project is not finished after one execution of the workflow. Instead, several iterations are necessary until a software for an ECU is completely finished.

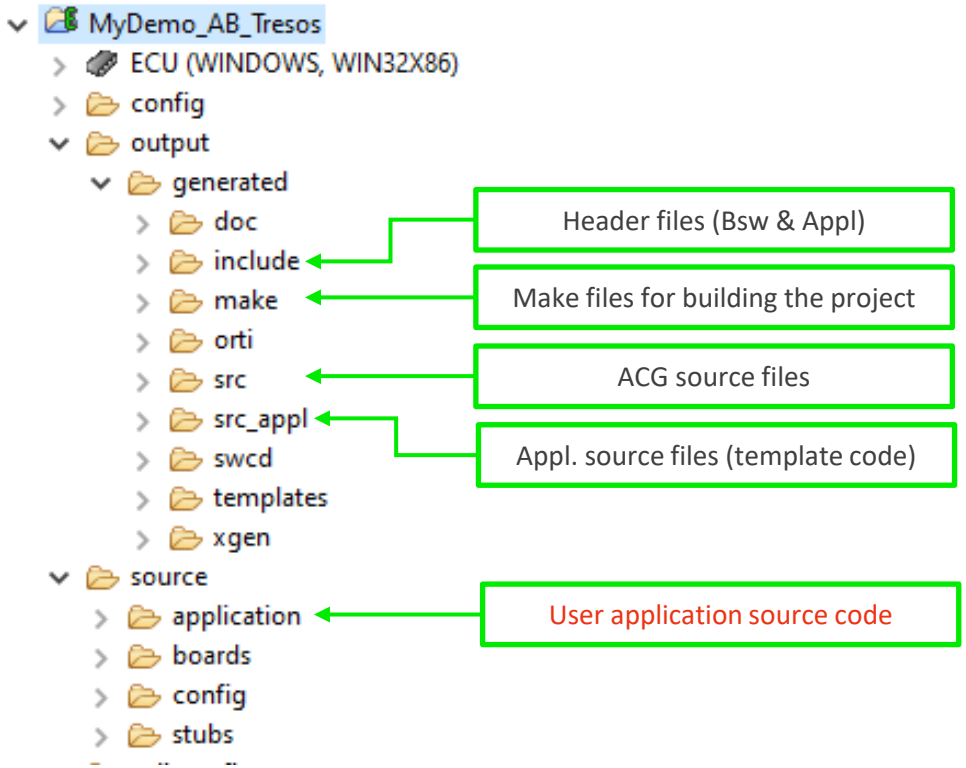
Depending on the changed input, one can jump back to dedicated workflow steps to iterate through the workflow:

- Changed basic software modules → jump back to slide 10
- Changed application → jump back to slide 17
- Revalidation in AUTOSAR Builder necessary → jump back to slide 19
- Additional runnable in SWC → jump back to slide 33
- Changed OS(Task) configuration → jump back to slide 40
- Changed RTE configuration → jump back to slide 42

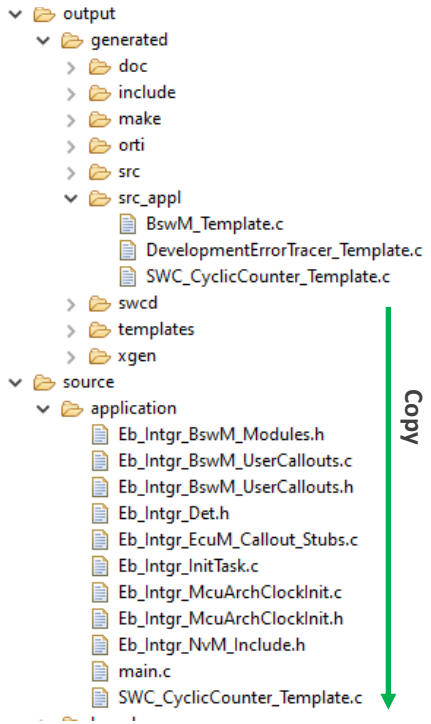
Generate source code and compile

Import System into RTE

1. **Generate code**
   - Run the EB tresos Studio code generation process
     - **Note**: You can ignore the warnings
- Inspect generated code
  - The generation process will output several files within your project



2. **Copy application source templates**
   Because the template code will be overwritten every time a generation process is started, we need to copy our "**SWC_CyclicCounter_Template.c** to our "User application source code" folder".



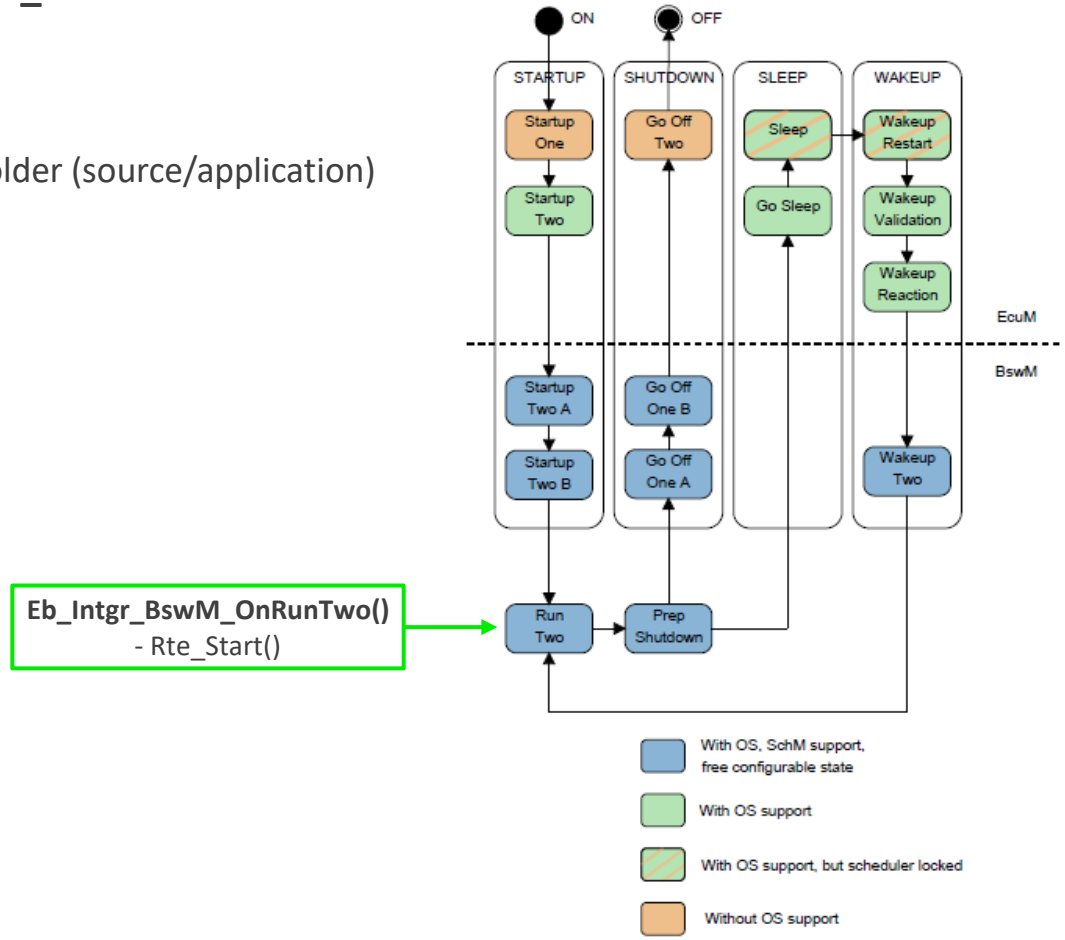
Update source code

Because the "basicTemplate" only does common **BSW scheduling** (without application) and we are now doing **Full scheduling** (with application)**,** we must start the RTE at the **RUN_TWO** state.

3. Enable Rte_Start

- Edit file "**Eb_Intgr_BswM_UserCallouts.c**" in "User application source code" folder (source/application)
- Search for function "**Eb_Intgr_BswM_OnRunTwo**"
- Remove following if/endif or change to #if 1 to **enable Rte_Start()** :

```
~~#if 0~~
 if ( Rte_Start() != E_OK )
 {
    /* Rte start failed */
 }
~~#endif /* 0 */~~
```



Compile

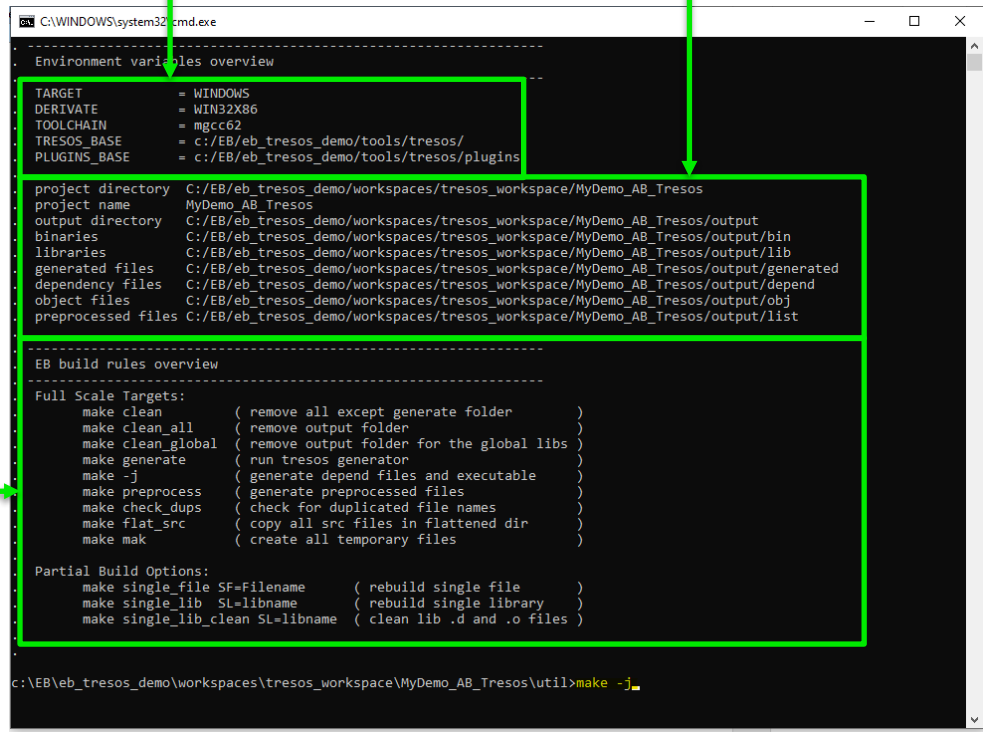
In this step we just want to compile our project. The "SWC_CyclicCounter_Template.c" does not have any logic – it only contains rudimentary source code for each runnable (C function).

1. Update **launch_cfg.bat**
   1. Edit lauch_cfg.bat in **<project>/util** folder
   2. Set the %TRESOS_BASE% variable to your EB tresos installation folder e.g.
      - IF [**%TRESOS_BASE%**]==[] SET **TRESOS_BASE=c:\EB\eb_tresos_demo\tools\tresos**
2. Run **launch.bat** and execute the following make statements
   1. make clean
   2. make –j

Environment variables

Project output structure

EB make rules



Note: On first compilation process, the make environment will automatically extract the GCC compiler.

Contact us

inquiries@elektrobit.com
elektrobit.com