

EB tresos classic AUTOSAR training

- Exercises



Elektrobit



Content

- Exercise description document
- Used tools
- Goal of the Exercises
- List of Exercises
 1. Getting started and create an EB tresos Studio project
 2. SWC modeling with Artext
 3. System Description Import
 4. Os and RTE Event Mapping
 5. RTE Service Port Mapping
 6. Com Stack
 7. Mode Management
 8. Memory Stack
 9. IO Hardware Abstraction example

Exercise description

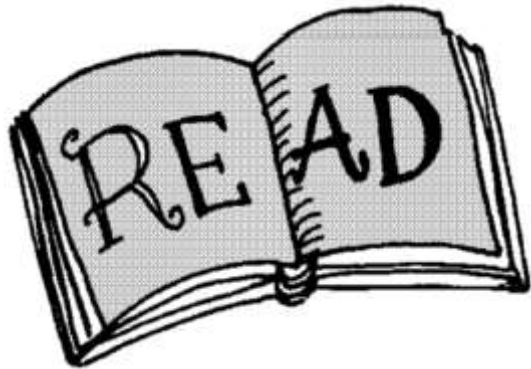
- Located in the installation directory → Subfolder slides

Local Disk (C:)		Name	Date modified	Type	Size
▼	EB_Training				
>	.metadata				
>	7-Zip				
>	AddStuff				
>	artext				
>	Eclipse_4_7_ApplicationDevelopment				
>	EclipseProjects				
	slides				
>	tresos_Inspector_and_BusmirrorRuntime				
▼	tresos_Wincore				
>	autosar				
	bin				
>	checksums				
>	configuration				
>	demos				
		00_Agenda_EN.pdf	04.03.2021 15:28	PDF Document	340 KB
		11_Introduction_&_Concepts_EN.pdf	04.03.2021 15:28	PDF Document	1.159 KB
		12_Architecture_&_Methodology_EN.pdf	04.03.2021 15:29	PDF Document	1.697 KB
		17_TresosStudio_EN.pdf	04.03.2021 15:29	PDF Document	2.088 KB
		21_BSW_Os_EN.pdf	04.03.2021 15:29	PDF Document	1.656 KB
		22_SWCLayer_SWCInternalsAndRTE_EN.p...	04.03.2021 15:30	PDF Document	2.400 KB
		24_Diagnostic_EN.pdf	04.03.2021 15:30	PDF Document	1.081 KB
		31_BSW_Com_EN.pdf	04.03.2021 15:31	PDF Document	1.671 KB
		32_BSW_Mode_EN.pdf	04.03.2021 15:31	PDF Document	1.402 KB
		33_BSW_Memory_EN.pdf	04.03.2021 15:34	PDF Document	909 KB
		34_Security_EN.pdf	04.03.2021 15:35	PDF Document	1.339 KB
		35_Safety_EN.pdf	04.03.2021 15:35	PDF Document	717 KB
		40_Exercise_Description.pdf	04.03.2021 15:36	PDF Document	2.346 KB
		41_Poster.pdf	04.03.2021 15:36	PDF Document	63 KB
		42_Poster-SysD-Bswmd-EcuC.pdf	04.03.2021 15:36	PDF Document	378 KB

Slide titles

Overview

- Overview slides give you an overview of the next exercise step
- no work is associated – just read....



Doing

- Configuration work is necessary follow the instructions which are described

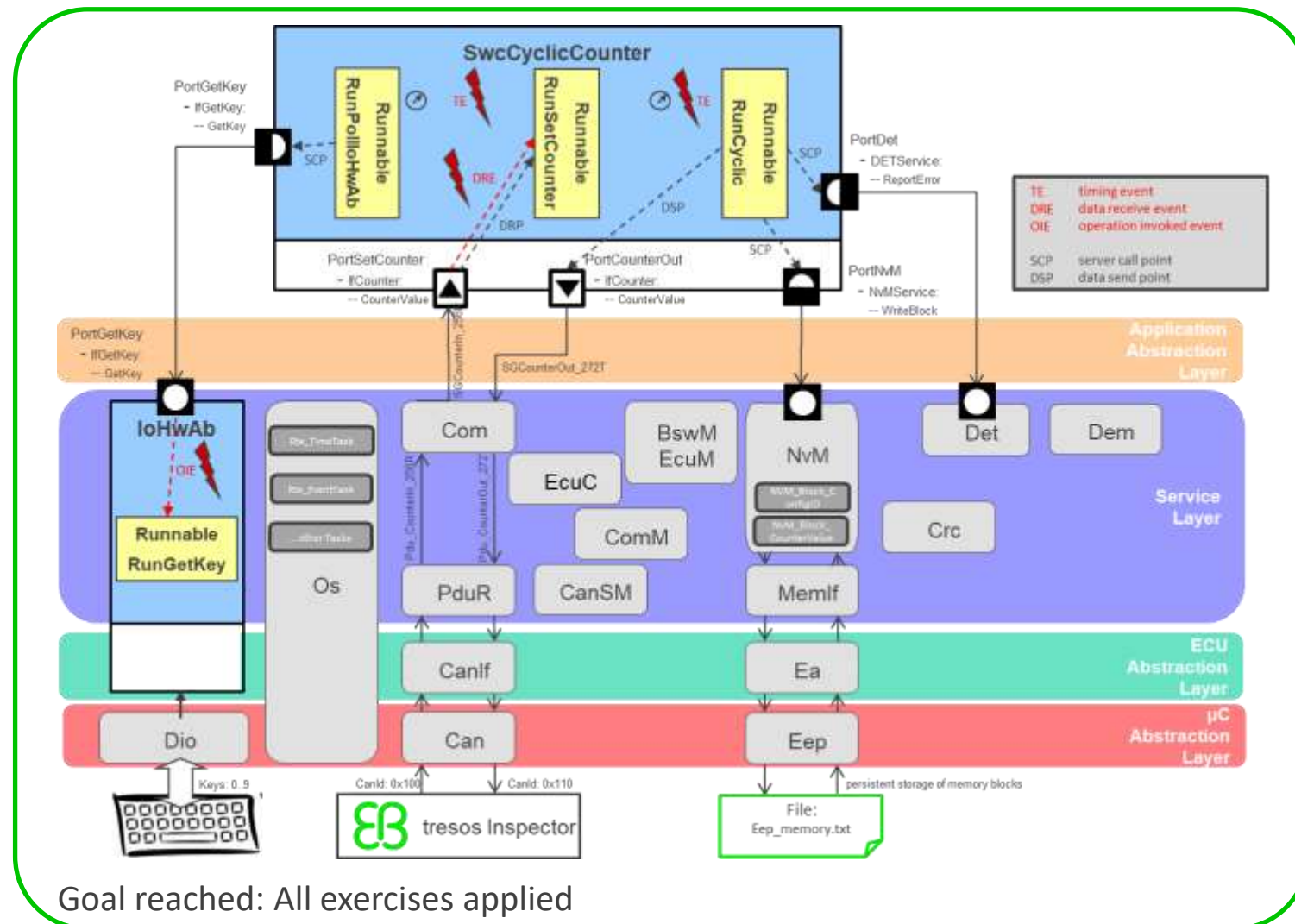
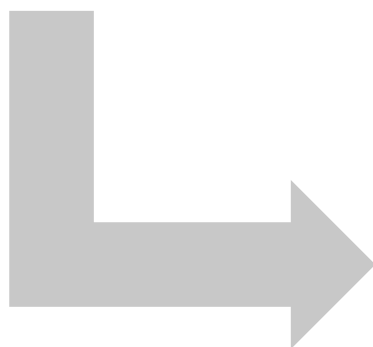
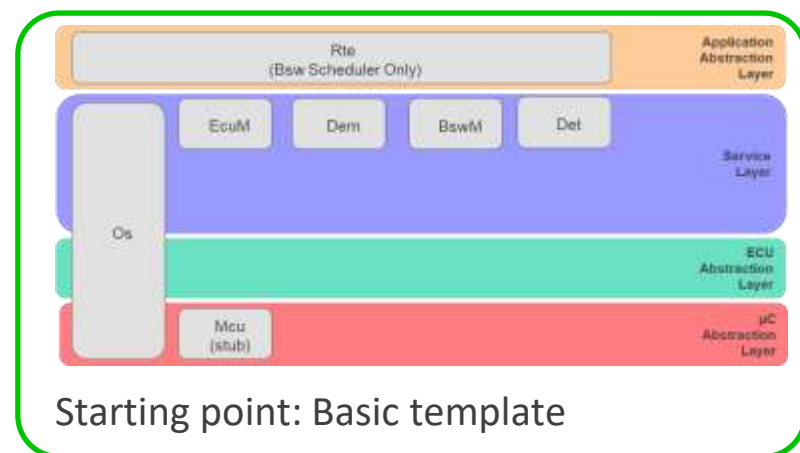


Accomplishments

- Summary of the accomplishments from the previous exercise step



Goal of the exercises



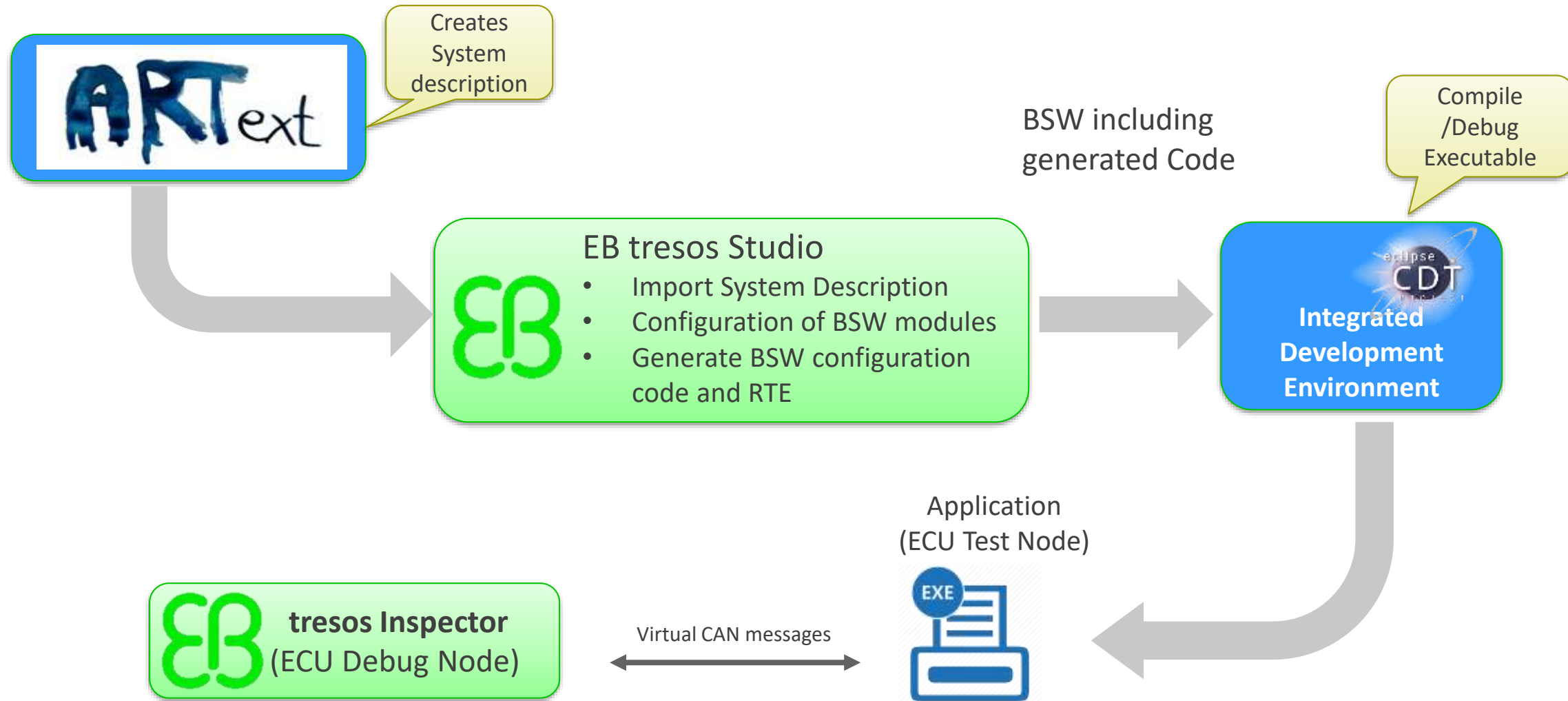
Overview: Used Tools



Elektrobit



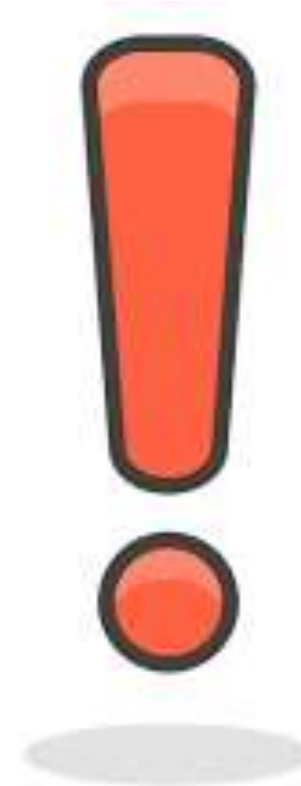
Overview Used Tools



Disclaimer about the EB tresos Inspector

- The tool “EB tresos Inspector” is not maintained anymore at EB
- We use the tool for the purpose of demonstrating the Com Stack exercise but it is fully optional (not mandatory)
- Support requests for EB tresos Inspector cannot be processed by EB anymore

If you wish to use EB tresos Inspector for the Com Stack exercise, please refer to the installation instructions



Exercise 1: Getting started and create an EB tresos Studio project



Elektrobit





- Overview:

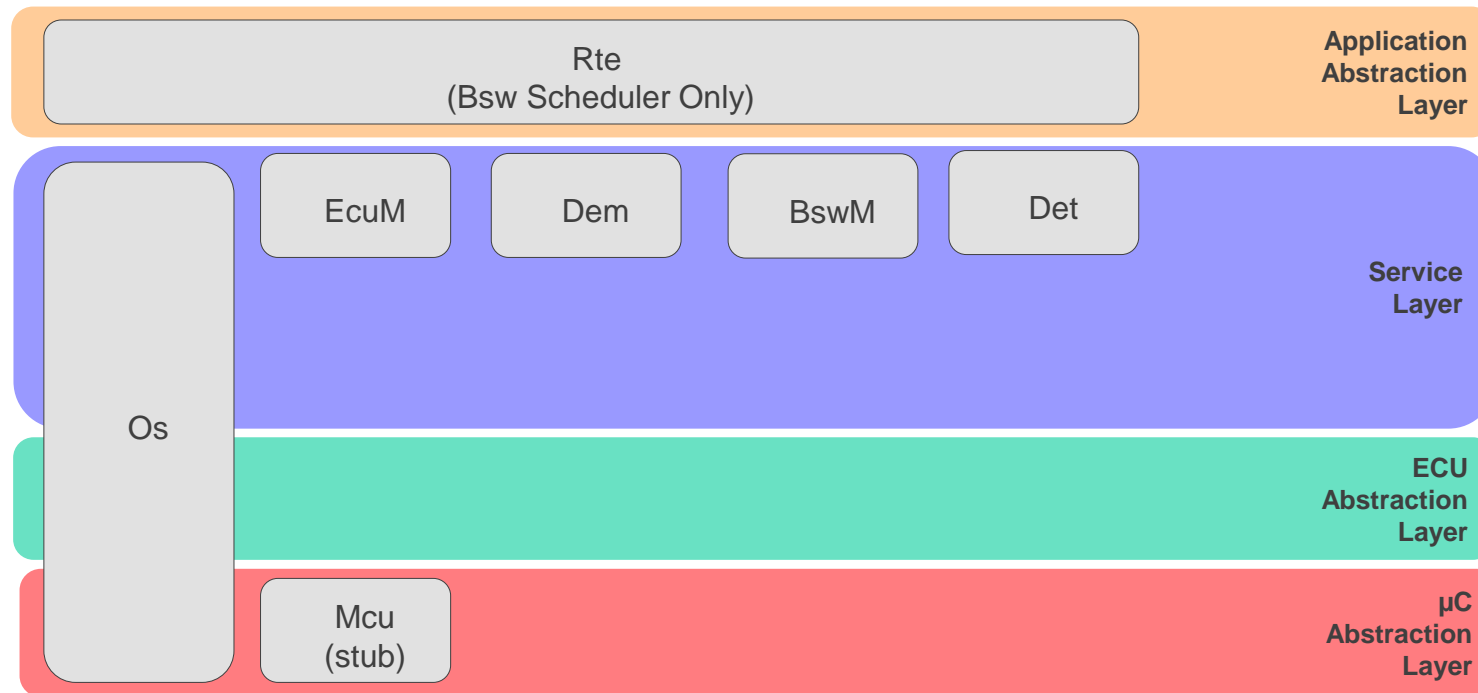
- This exercise step uses a so called basic template to set up a configuration project which includes mandatory and preconfigured BSW modules

- Objectives:

- Get familiar with the EB tresos Studio
- Create a project based on a template
- Inspect GUI and navigation
- Generate code
- Use CDT to compile / debug the application



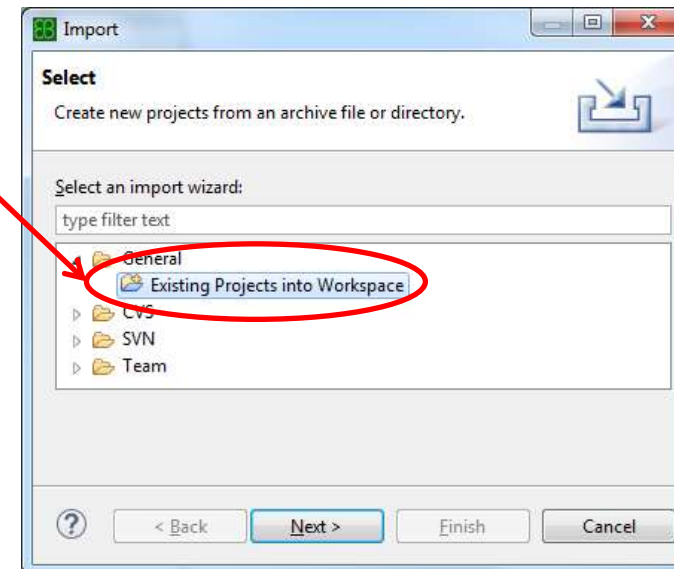
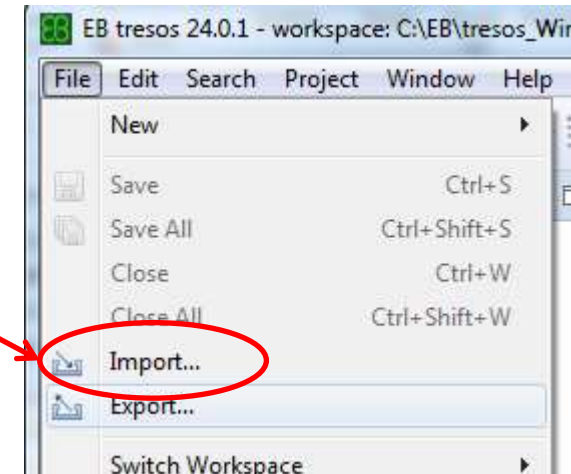
The Basic Template:





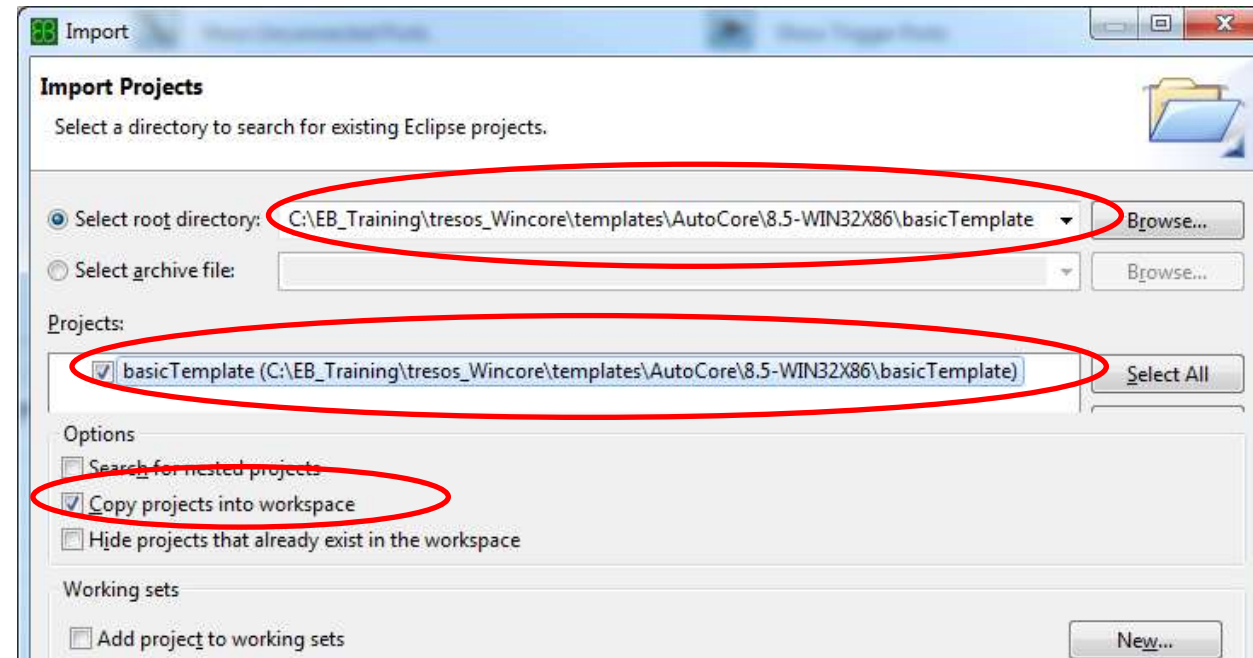
Start C:\EB_Training\StartTresosStudio.bat

- Click „Yes“ if asked to create workspace
- Use Menu -> „File“ -> „Import...“
- Select General/Existing Projects into Workspace






Import Template



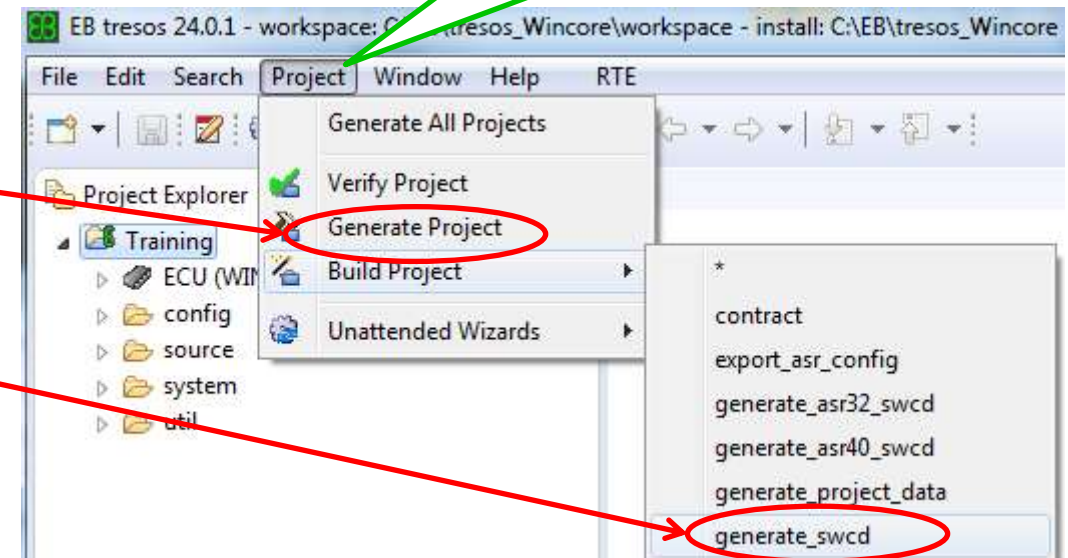
- Select root directory
C:\EB_Training\tresos_Wincore\templates\AutoCore\8.5-WIN32X86\
- Check basicTemplate
- Check „Copy projects into workspace“
- „Finish“



Rename and Generate

- Rename the project using F2 or context menu
 - Rename to „Training“
- Double-click on  ECU (WINDOWS, WIN32X86) to load the Ecu configuration
- Trigger code generation for the new project
 - Generate Project: Generates BSW module code based on the current configuration
 - generate_swcd: Generates basic software component descriptions of the BSW based on the current configuration.

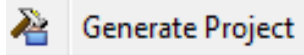
Code generation can be triggered from context menu (right-click on project) or via the toolbar



Getting started with EB tresos Studio – Accomplishments (1)

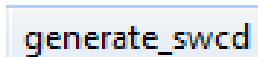


- At this phase of the exercise, you accomplished the following:
 - You created a project in EB tresos Studio which is based on a pre-configured template
 - You generated the configuration depend source code for each BSW module which is part of the project



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

- You generated the basic software module description files



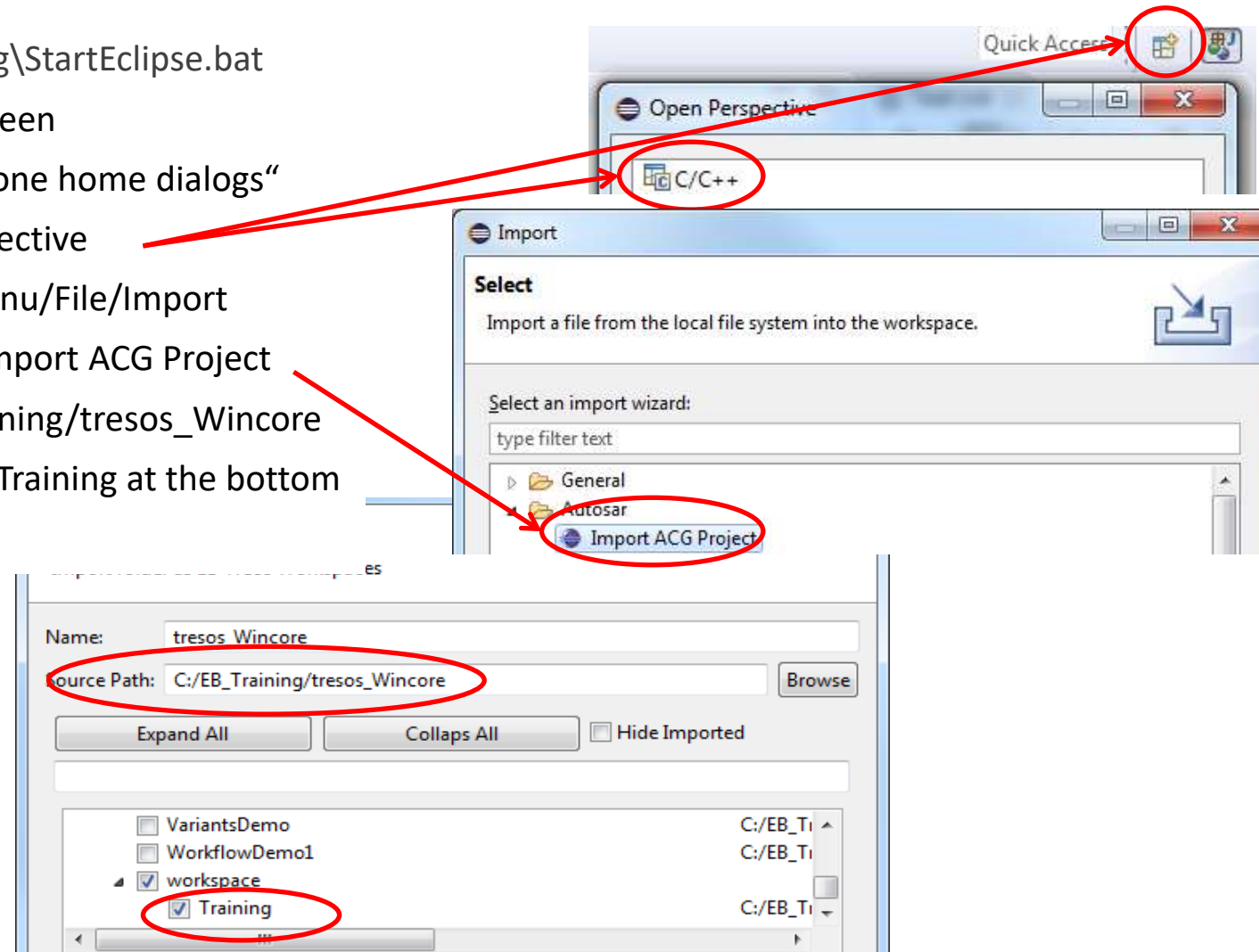
The generated files are located in the folder [project]\output\generated\swcd

Note: While the generated BSW module code is required for compilation, the basic software component descriptions are needed for updating the System Model and will be used in the following exercise steps!



Start the Eclipse CDT (Build & Debug environment)

- Run C:\EB_Training\StartEclipse.bat
- Close welcome screen
- Close all „want phone home dialogs“
- Open C/C++ perspective
- Import Project Menu/File/Import
- Choose Autosar/Import ACG Project
- Browse C:/EB_Training/tresos_Wincore
- Check workspace/Training at the bottom
- Finish

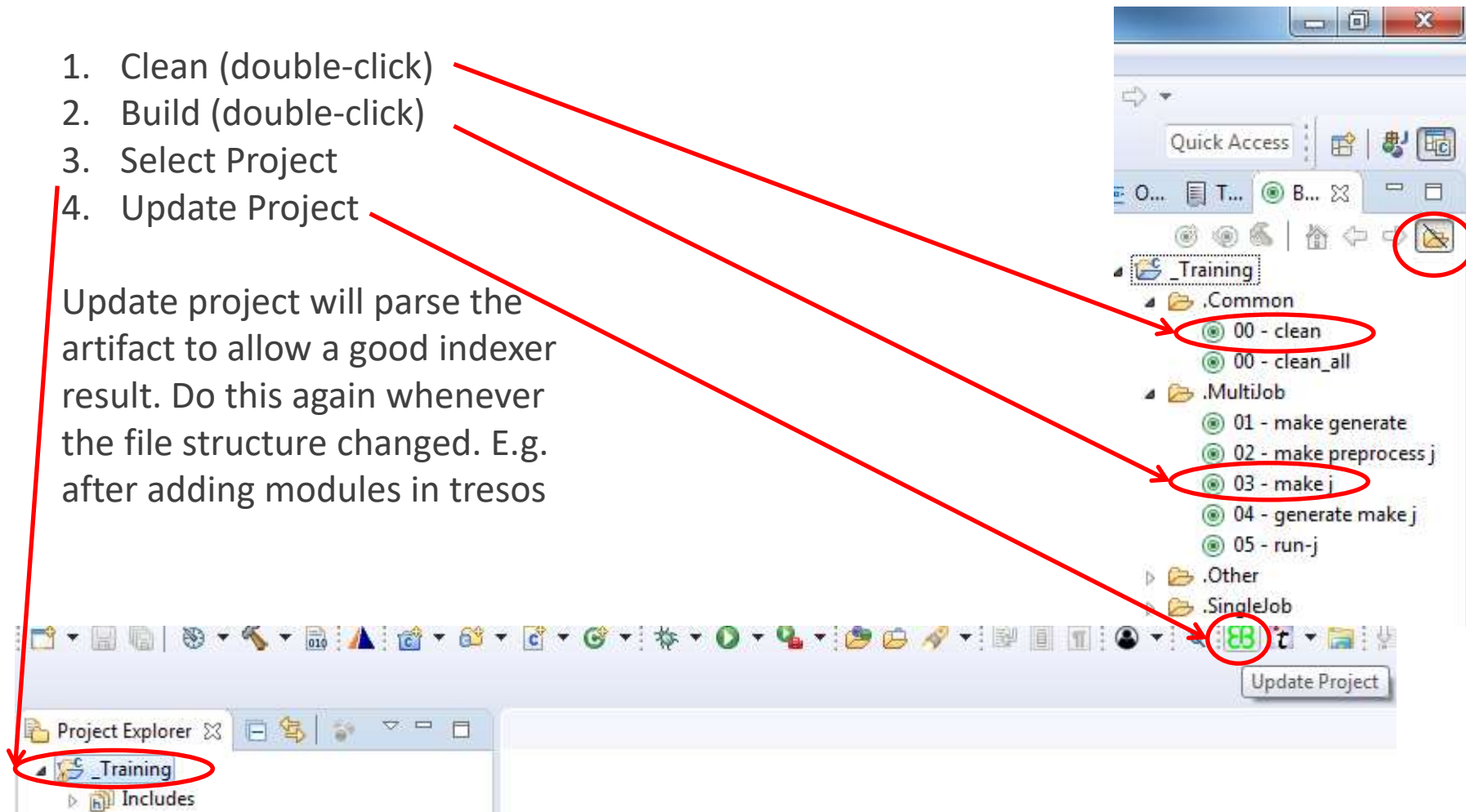




Build the project

1. Clean (double-click)
2. Build (double-click)
3. Select Project
4. Update Project

Update project will parse the artifact to allow a good indexer result. Do this again whenever the file structure changed. E.g. after adding modules in tresos





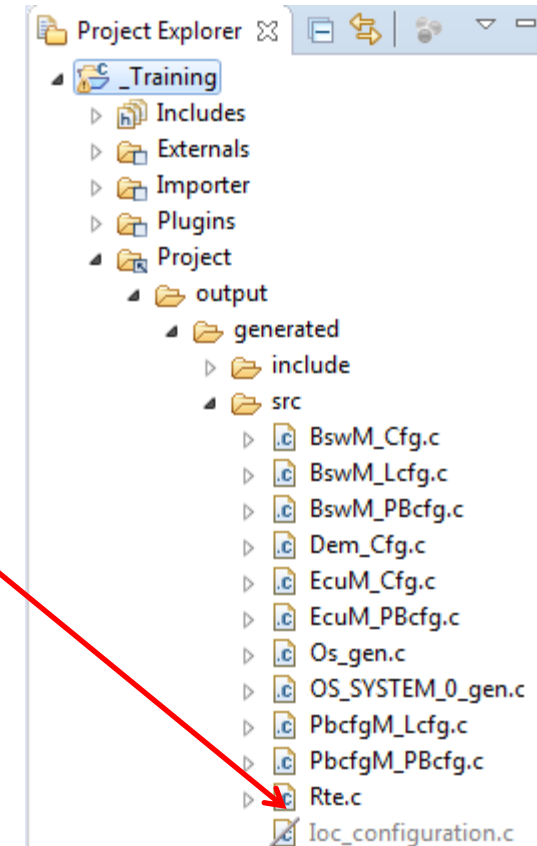
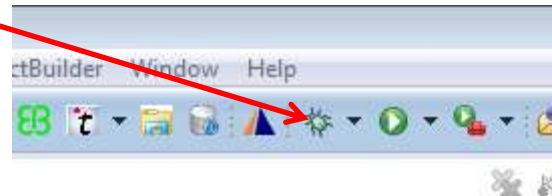
Start Debugging

1. Open Rte.c
2. Set breakpoint at shown Task

```
97 TASK(SchMDiagStateTask_20ms)
98 {
99     /* scenario A1/B1 (A1) */
100     Rte_Task_Dispatch(SchMDiagStateTask_20ms);
101     {
102     }
103     Rte_Schedulable_EcuM_MainFunction_Start();

```

3. Start debugging:
4. Debug



Getting started with EB tresos Studio – Accomplishments (2)



- At this phase of the exercise you accomplished the following:
 - You validated that the toolchain is working
 - Eclipse is used for building and debugging the application
 - You observed that the basic template based project is actually compilable, executes and has at least one cyclically activated task already
- Outlook: You will learn more about the mechanisms behind the scheduling of BSW main functions in one of the following exercises

Exercise 2: SWC modeling with Artext



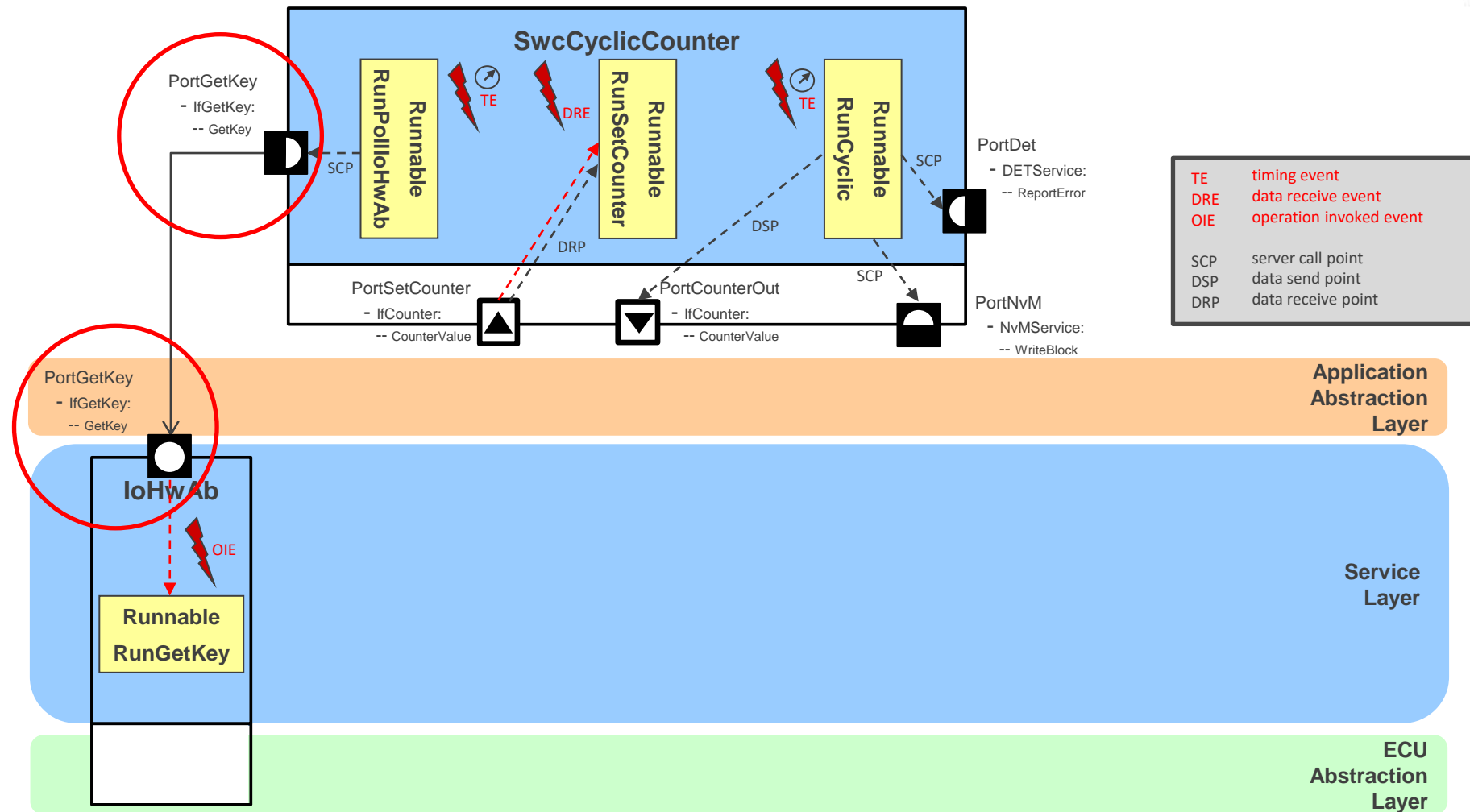
Elektrobit





SWC modeling - Objectives

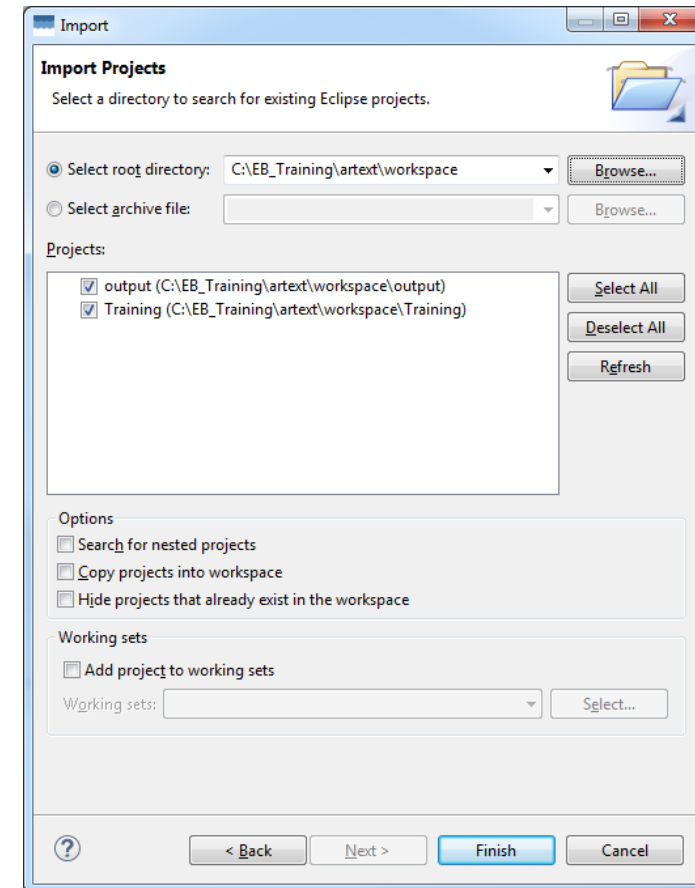
- Working on a SWC description with an AUTOSAR Authoring tool
- Model a client/server interface that can be used in the application
- Apply this client/server interface to a new port on the SWC “SwcCyclicCounter” (acting as a client)
- Connect the port of the SwcCyclicCounter to the corresponding providing port on the “IoHwAb” component
- Export the model from Artext to AUTOSAR arxml





Open Artext and load existing projects into workspace

- Open ArText C:\EB_Training\StartArtext.bat
- Import all existing projects in the workspace
- Open Training project → folder CounterDemo
- There are 4 separate .swcd files:
 - CounterDemo.swcd
 - IoHwAb.swcd
 - PortInterfaces.swcd
 - SwcCyclicCounter.swcd
- Remark: The following files have been added to the project to allow Artext to resolve the modes/interfaces which are referenced in the swcd files:
 - BswMModes.arxml
 - ServiceInterfaces.arxml





Define a new Client/Server interface

- In PortInterfaces.swcd add the new interface:

```
interface clientServer service IfGetKey {  
    operation GetKey {  
        out uint8 GetKeyValue  
    }  
}
```

- Hint: When typing the additional code, use “Ctrl” key + “space” to get context specific code completion proposals
- Of course you can also copy & paste the code section above



Add a client port and connect it

- In SwcCyclicCounter.swcd add **the marked lines** to define the client port:

```
component application SwcCyclicCounter {  
    ports {  
        ...  
        client PortGetKey requires IfGetKey  
        client PortDet requires DETService  
        client PortNvM requires NvMService  
    }  
}
```

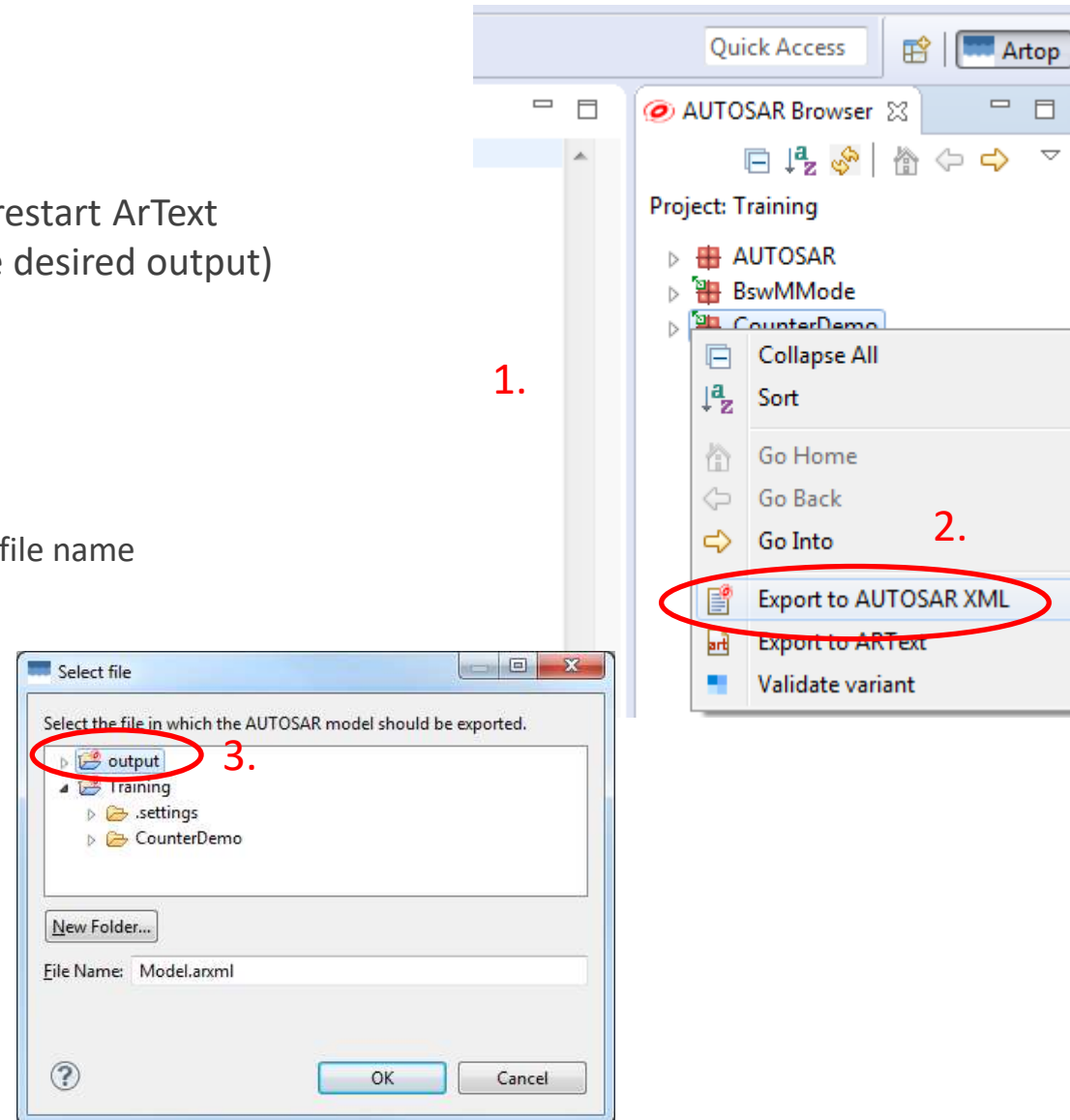
- In CounterDemo.swcd add **the marked lines** to connect the P-Port to the R-Port:

```
composition TopLevelComposition {  
    prototype SwcCyclicCounter SwcCyclicCounterInstance  
    prototype IoHwAb IoHwAbInstance  
    connect IoHwAbInstance.PortGetKey to SwcCyclicCounterInstance.PortGetKey  
}
```



Export

- Save
- If some problem markers do not disappear, please restart ArText before exporting (otherwise it will not generate the desired output)
- Export to AUTOSAR XML (arxml file)
 - Select CounterDemo packages in AUTOSAR Browser:
 - right click and „Export to AUTOSAR XML“
 - Select output project for export – keep the proposed file name (Model.arxml)



SWC modeling– Accomplishments



- At this phase of the exercise, you accomplished the following:
 - You used a AUTOSAR Authoring to model properties of a SW-C
 - For training purposes, the tool “Artext” was used which is not a commercial tool but rather a tool demonstrator for the tooling platform developed by the Artop consortium
 - You exported the Software Component Description to AUTOSAR XML so it can be used by EB tresos Studio → next step of the exercise

Exercise 3: System Description Import



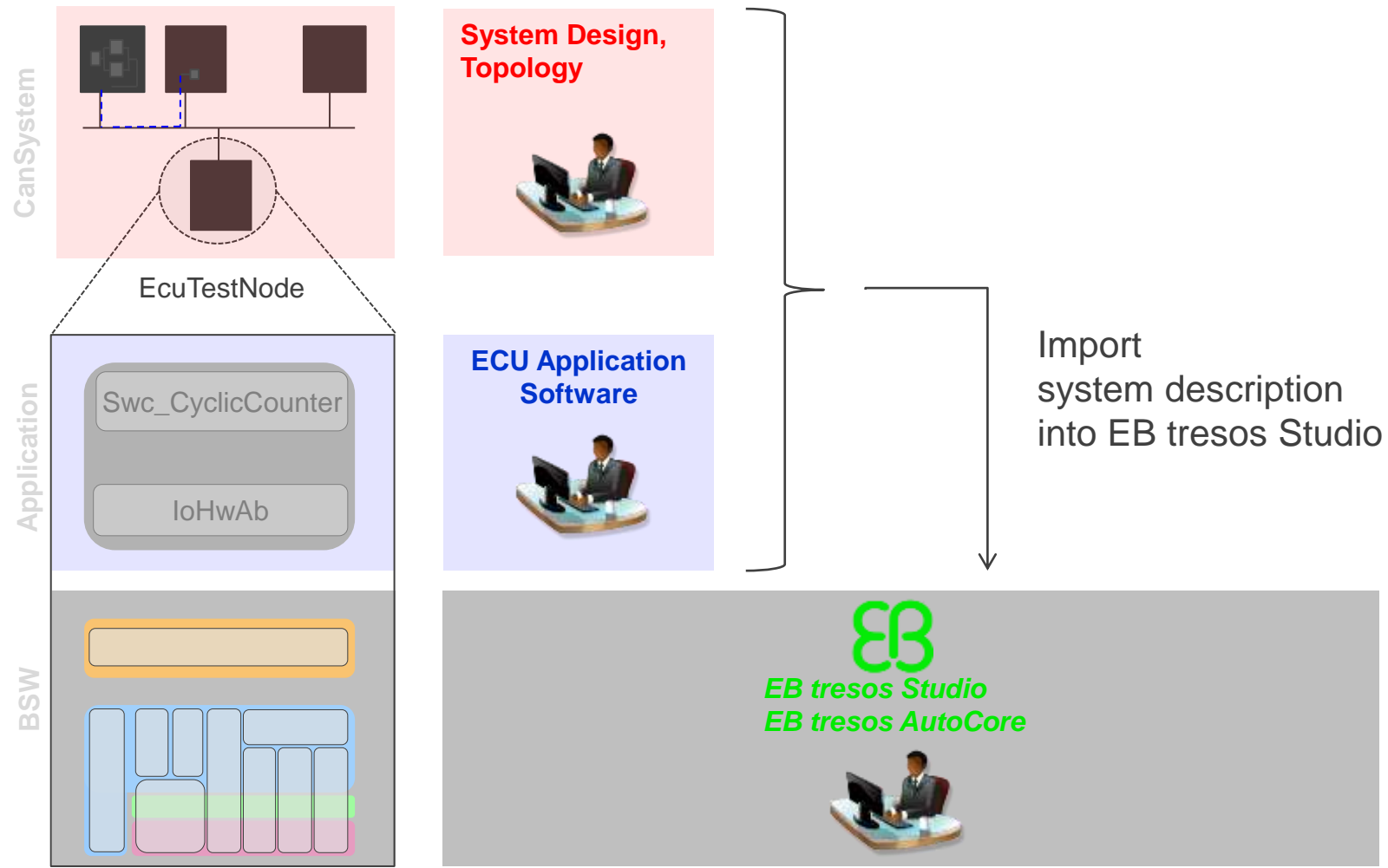
Elektrobit





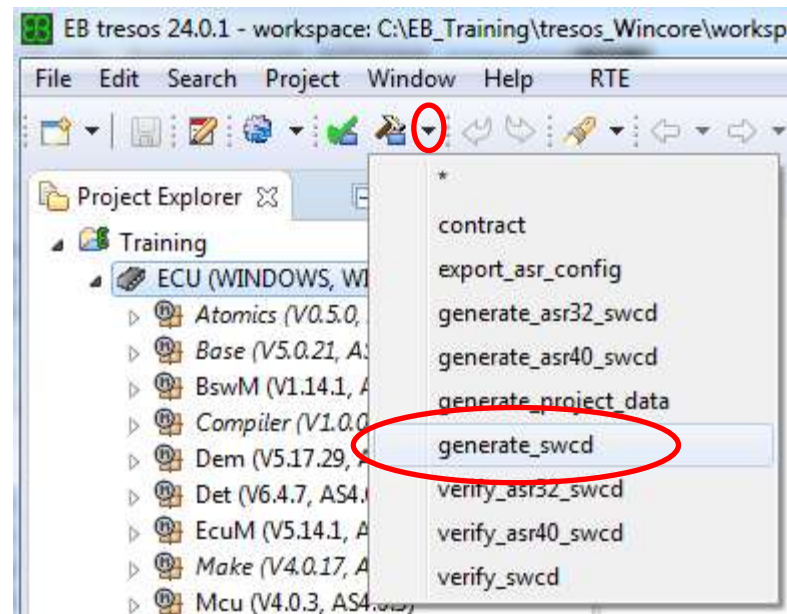
System Description Import - Objectives

- Use the System Description Importer in EB tresos Studio
 - Define Importer settings
 - Understand which files are required for the import and how the importer works
 - Run the System Description importer
 - Inspect the result after the import:
 - System Model Viewer
 - Rte editor
 - Composition Editor





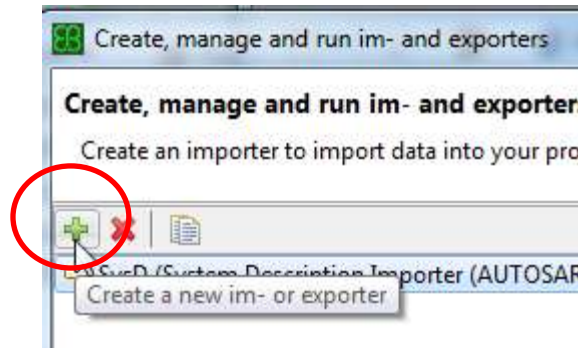
- Start tresos Studio and load the Training project.
- Run generate_swcd (if not yet done in the previous part of the exercise)
 - Rationale: The files generated by tresos will be needed for the System Description Import!





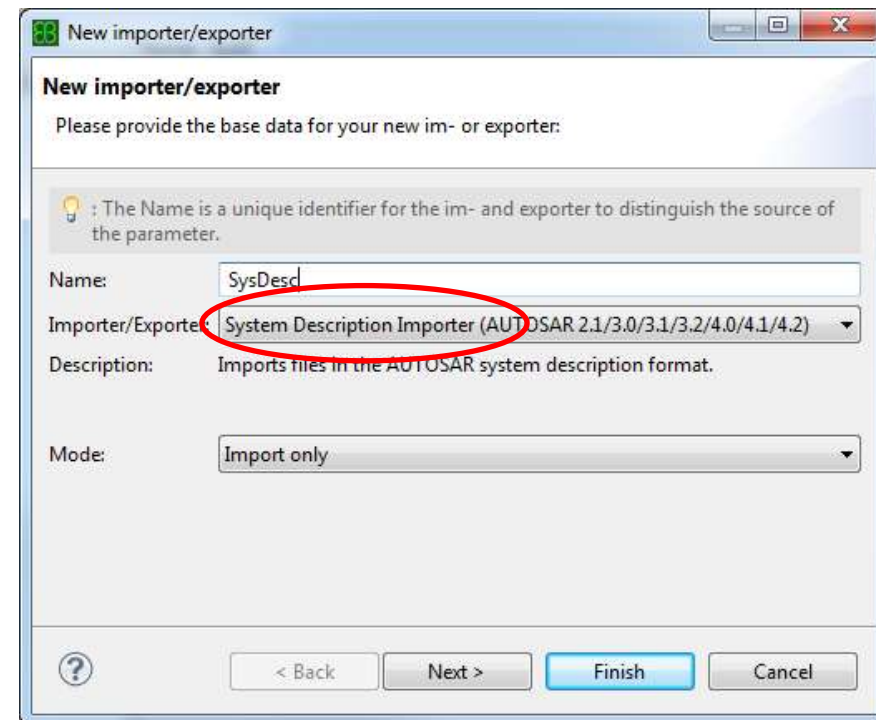
Create Importer

- In “Project Explorer”, right click on project “ECU (WINDOWS, WIN32X86)”
 - Select “Im- and Exporters”
- Add an Importer



- Choose System Description Importer and Name it e.g. “SysD”

→ Next





Configure importer

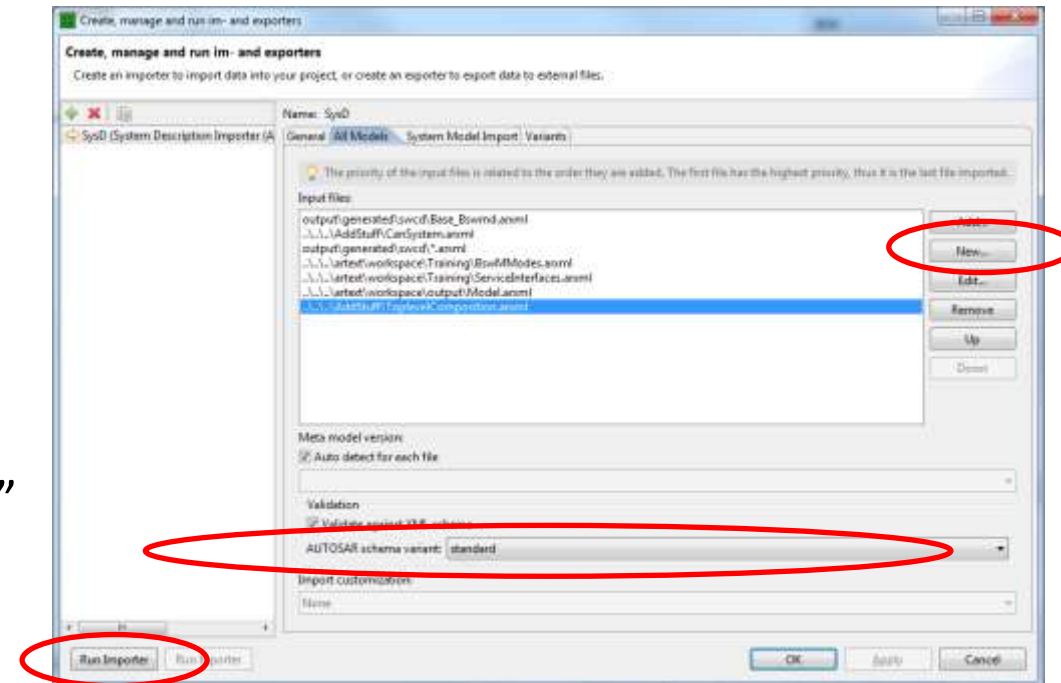
- Use „New...“ to add the following entries:

```
output\generated\swcd\Base_Bswmd.xml
..\..\..\AddStuff\CanSystem.xml
output\generated\swcd\*.xml
..\..\..\artext\workspace\Training\BswMModes.xml
..\..\..\artext\workspace\Training\ServiceInterfaces.xml
..\..\..\artext\workspace\output\Model.xml (*)
..\..\..\AddStuff\ToplevelComposition.xml
```

(*) If you skipped the “SWC modeling with Artext” exercise, you may also use

```
..\..\..\AddStuff\solution\Model.xml
```

- Set schema variant to „standard“
- → Next
- Note that the importer uses “Overwrite existing model”
- → Finish
- → Apply
- → Run Importer



System Description Import – Accomplishments (1)



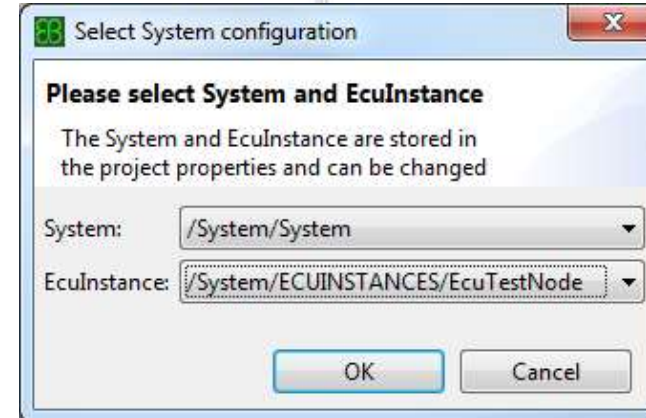
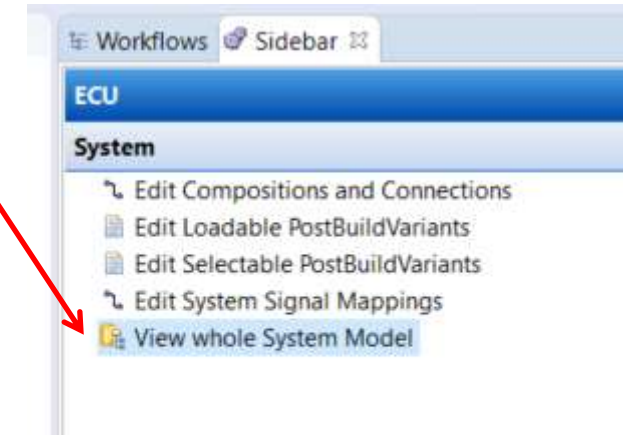
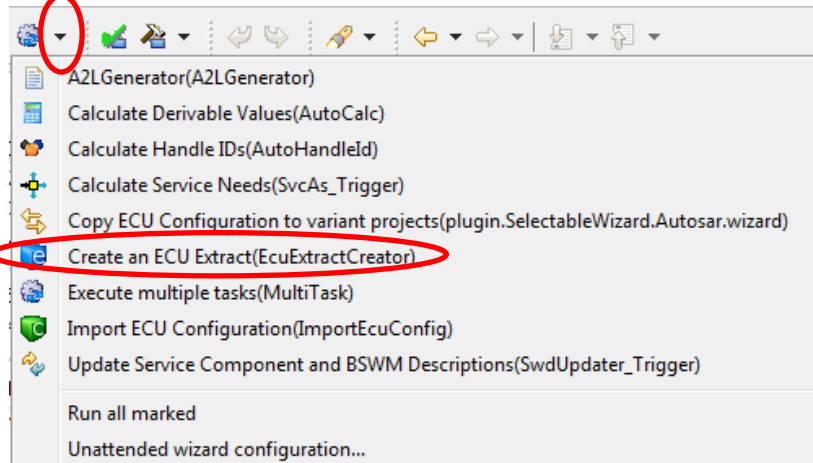
- At this phase of the exercise you accomplished the following:
 - You configured an “System description importer” in your project with the following input files
Note: all path names are referring to the relative location of the project

Input File	Content (reason of order)
output\generated\swcd\Base_Bswmd.arxml	Should be very first file to make sure the platform base types are used and not any other that might be wrong/incomplete.
..\..\..\AddStuff\CanSystem.arxml	System / Topology description: Should always be 2nd after base_bwmd. in arxml the system part is not completely splittable. this might lead to problems if other system fragment files are imported before.
output\generated\swcd*.arxml	The generated SWCDs from the Basic Software modules (generate_swcd) Reason: tresos does not automatically merge these descriptions and it is recommended to import these descriptions always with the System Description
..\..\..\artext\workspace\Training\BswMModes.arxml	Service Module Interfaces used by the application and must therefore be available for the Importer to resolve references
..\..\..\artext\workspace\Training\ServiceInterfaces.arxml	
..\..\..\artext\workspace\output\Model.arxml	The export of the modeled SWC Descriptions from Artext
..\..\..\AddStuff\ToplevelComposition.arxml	The mandatory Top Level Composition



Inspect results after the System Description Import (1)


- Use “View whole System Model” to inspect the result after the import
- Run "Create an ECU Extract" wizard. Doing this the first time, tresos will ask you to select the System and ECU Instance

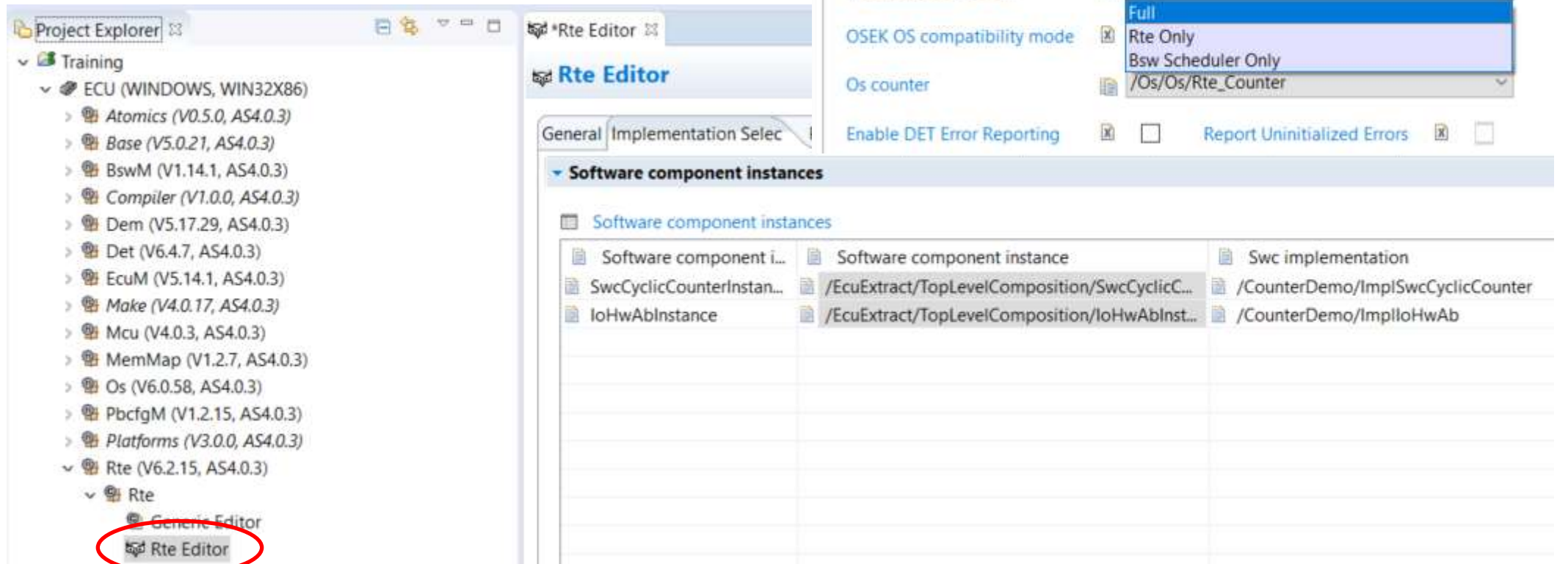


- Use “View whole System Model” again to inspect the result after the using the ECU Extract wizard
 - Hint: There will be a new package “EcuExtract”



Inspect results of using the System Description Importer (2)

- Start “Rte Editor” 
 - In tab General set “Rte Generator Output” to “Full”
 - Now inspect the tabs “Implementation Selection”



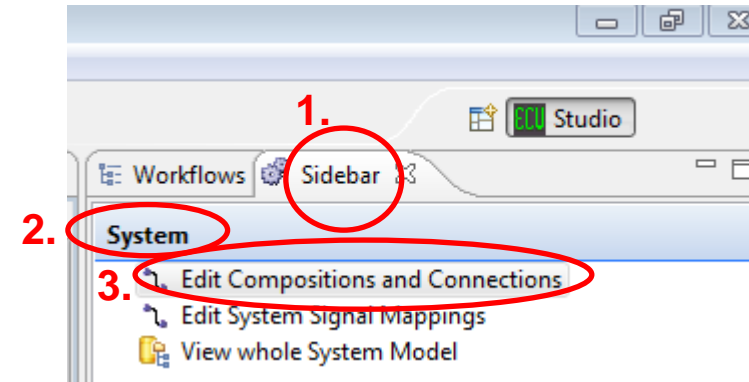
The screenshot displays the Rte Editor interface. On the left, the Project Explorer shows a tree structure under 'Training' > 'ECU (WINDOWS, WIN32X86)'. The 'Rte' component is expanded, and the 'Rte Editor' sub-component is highlighted with a red circle. The main window shows the 'Rte Editor' tab with the 'General' sub-tab selected. The 'Rte Generator Output' dropdown is set to 'Full'. The 'OSEK OS compatibility mode' dropdown is set to 'Rte Only'. The 'Os counter' dropdown is set to '/Os/Os/Rte_Counter'. The 'Enable DET Error Reporting' checkbox is checked, and the 'Report Uninitialized Errors' checkbox is unchecked. Below these settings, the 'Software component instances' table is visible.

Software component i...	Software component instance	Swc implementation
SwcCyclicCounterInstan...	/EcuExtract/TopLevelComposition/SwcCyclicC...	/CounterDemo/ImplSwcCyclicCounter
IoHwAbInstance	/EcuExtract/TopLevelComposition/loHwAbInst...	/CounterDemo/ImplIoHwAb



Inspect results of using the System Description Importer (3)

- Close the Rte editor
- Start Connection editor in the Sidebar
 - Inspect the imported system
- Close Connection editor
 - Do not save to system model as there are no changes



Entity	Target	Interface
TopLevelComposition		
IoHwAbInstance		
PortGetKey		IfGetKey
IoHwAbInstance_PortGetKey	SwcCyclicCounterInstance/PortGetKey	
SwcCyclicCounterInstance		
PortCounterOut		IfCounter
PortDet		DETSERVICE
PortGetKey		IfGetKey
IoHwAbInstance_PortGetKey	IoHwAbInstance/PortGetKey	
PortNvM		NvMService
PortSetCounter		IfCounter

System Description Import – Accomplishments (2)



- At this phase of the exercise you accomplished the following:
 - You used the ECU extract Wizard which creates an ECU Extract from the tresos internal System Model (refer to training chapter “tresos Studio” for more background information)
Note: The ECU Extract is the required input for the RTE (Editor and Generator) and the RTE Editor will complain about a missing ECU Extract if you try to run it without executing the ECU Extract Wizard before
 - Remember that the ECU Extract wizard has to be executed every time the EB tresos internal System Model is updated
 - You observed elements of the imported System Description in the following views in EB tresos Studio:
 - System Model Viewer
 - RTE Editor, Tab “Implementation Selection” lists the imported SWC Types and their mapping to an implementation
 - Connection editor: This editor shows all the Ports from the imported SWCD and allows to map them to Service ports. This mapping will be done in subsequent exercise steps

Exercise 4: Os and RTE Event Mapping



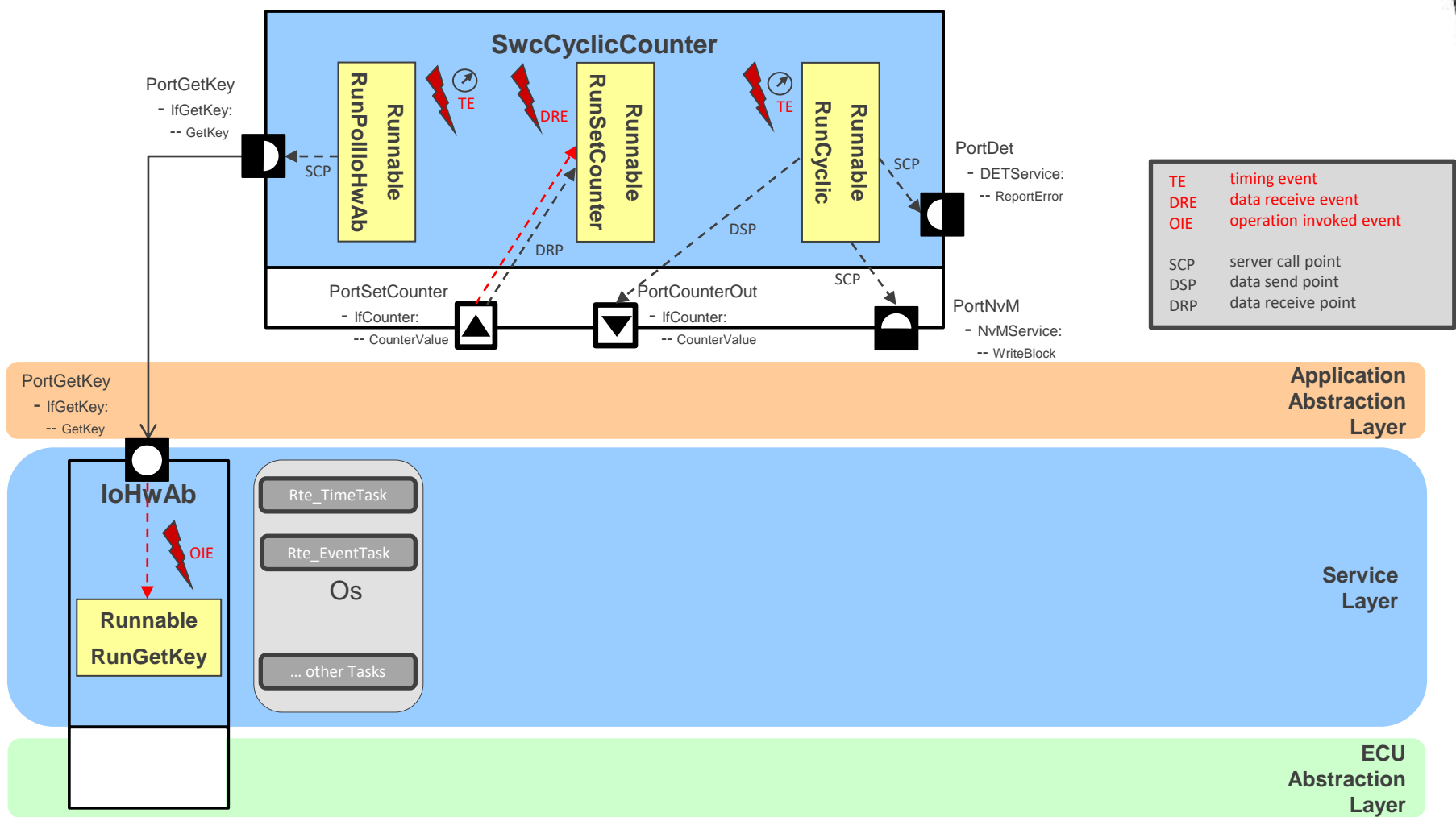
Elektrobit





Os and RTE Event Mapping - Objectives

- Objective :
 - Configure the Os / Rte to enable the execution of the Application SWC “SwcCyclicCounter” in the previous exercise step
- This exercise step consists of
 - Adding Os tasks for our Training project which will be used for the application
 - Map BSW Events to Tasks (automatically) - not be required at this step of the exercise
 - Map Rte Events to Tasks (manually)
 - Run "Calculate Service Needs" wizard
 - Build project
 - Run / Debug the application





Os Editor – New tasks for the application

- Open EB tresos Studio and open the Os editor
 - Switch to tab: „OsTask“
 - Add two tasks:
 - Rte_TimeTask (priority: 10)
 - Rte_EventTask (priority: 20)

Os (Os)

Os

Name Os

General EB Published Information OsAlarm OsAppMode OsApplication OsCounter OsEvent OsSpinlock Osloc OsLsr OsResource OsTask OsSc

OsTask

Index	Name	OsTaskActivation	OsTaskPriority	OsStac...	OsTask...
0	Init_Task	1	127	1024	NON
1	SchMDiagStateTask_20ms	1	50	512	FULL
2	Rte_time	1	10	1024	FULL
3	Rte_event	1	20	1024	FULL



Rte Editor – Event to task mapping

- Open Rte Editor
- Switch to tab “Event Mapping”
- 1. Auto-Map BSW events
- 2. Select all “RteTimingEvent”s
- 3. Select the Rte_TimeTask
- 4. Map the selected events to the selected task
- 5. Repeat the steps above (2-4) for the DataReceivedEvent and map it to the Rte_EventTask

Unmapped RTE and BSW events

Event	Executable e...	Period	Offset
TimingEvent (RteTimingEvent)	RunCyclic	500.0 ms	0.0 ms
TimingEventForIoHwAb (RteTimingEvent)	RunPollIoHwAb	50.0 ms	0.0 ms
DRESetCounter (DataReceivedEvent)	RunSetCounter		
OIEGetKey (OperationInvokedEvent)	RunGetKey		

4.

1.

3.

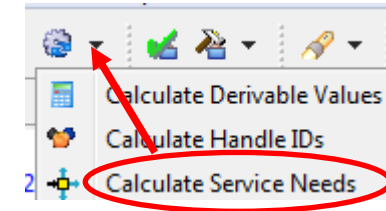
Task

/Os/Os/Rte_TimeTask



Calculate Service needs & enable Rte_Start

- Close Rte Editor and Run "Calculate Service Needs" wizard:
 - inspect results → The wizard updates the Os configuration according to the event mapping
- Generate project
 - Note: 12 warnings remain, among other things because of not yet connected ports




- Enable Rte_Start
 - Edit file C:\EB_Training\tresos_Wincore\workspace\Training\source\application\Eb_Intgr_BswM_UserCallouts.c" **line 819**
 - 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 */
```



Application & SW Build

- Add application implementation files to the project source folder
 - Copy the application files from “C:\EB_Training\AddStuff\swc” to “C:\EB_Training\tresos_WinCore\workspace\Training\source\application”
- Switch to eclipse CDT
 - Terminate previous debugging session
 - Build the project (see Tool exercise again), (clean, make j)
 - Debug the project:
 - Switch to Debug perspective
 - In Project Explorer, Select “_Training”, then execute by pressing 

Os and Rte Event Mapping - Accomplishments



- At this phase of the exercise, you accomplished the following:
 - You added and configured two tasks in the Os
 - You mapped RTE Events to the previously configured Os tasks
 - At this stage the only unmapped RTE events are those which were defined as part of the SWC Description and imported with the System Description importer
 - You used the “Calculate Service needs” wizard → It evaluates the Service needs of the RTE and translates them into configuration of modules which provide the “Service” – in this case the Os
 - Note: You can observe newly added configuration items in the Os Editor afterwards

Exercise 5: Rte Service Port mapping



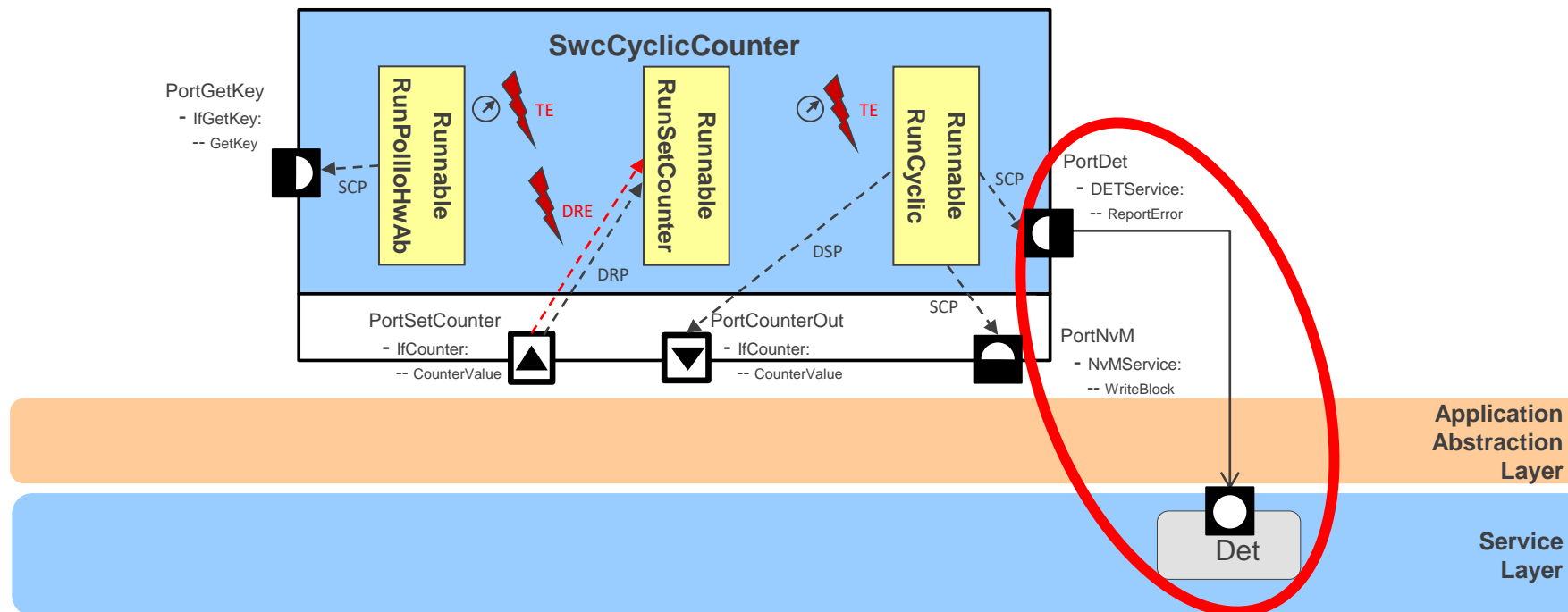
Elektrobit





Rte Service Port mapping Exercise - Objectives

- Configure a Service module (here: Det) to provide it's service to the application via a Service Port
- Add a instance of the service module to the TopLevelComposition
- Connect the service port to the compatible port of the application using the Composition and Connection Editor






Rte Service Port mapping Exercise - Doing

- Open Det module
 - In tab “Det Accessing Software Components”:
 - select “Enable RTE usage”
 - add one Component with Module ID 256 (default value)
 - In tab “Det Notifications”
 - Enable “DetNotification” (→ User callback)
 - Add “DetErrorHook” : “My_DetCallback”
 - Add “DetHeaderFile” : “My_DetCallback.h”
- Run „generate_swcd”
 - This step updates the basic software module descriptions – here needed because the Det now contains a service port
- Run System Description Import again to update the tresos System Model with the updated basic software module description
 - Hint: you can create your own multiple task wizards for repetitive tasks.



Rte Service Port mapping Exercise - Doing

- Open Connection Editor (Sidebar → System)
 - Note that the Det is not yet part of the TopLevelComposition
 - Add an instance of Det via „Add prototypes to TopLevelComposition“ 
 - Right click on Det port and „Add Connector“ to SwcCyclicCounterInstance → PortDet
- Close Connection Editor (Confirm „Save to system model“)

- Add the file „systemmod/ConnectionEditor.arxml“ to the System Importer
 - This step ensures that the changes done with the Connection Editor will be preserved during a subsequent System Description import

- Run Create Ecu Extract Wizard

- Open Rte Editor → Go to tab „Implementation Selection“ → Close Rte Editor
 - This step ensures that the RTE configuration is updated

- Run Calculate Service Needs Wizard



Rte Service Port mapping Exercise - Doing

- Generate code for the project
- Add Det user callback implementation
 - Copy the callback files from “C:\EB_Training\AddStuff\DetCallback” to “C:\EB_Training\tresos_WinCore\workspace\Training\source\application”
- Switch to eclipse CDT
 - Build project (according to previous exercises) (clean, make j)
 - Start execution in debugger (according to previous exercises)

Rte Service Port Mapping - Accomplishments



- At this phase of the exercise, you accomplished the following:
 - You configured a Service port in the Det module
 - You regenerated BSW module descriptions and run the System Description importer again
→ the EB internal System Model was updated with the updated descriptions of BSW modules
 - Based on the new import, you used „Create an ECU Extract“ to update the ECU Extract of the system model
 - You added an instance of the Det module and added this to the Top Level Composition.
This is a precondition for connecting the service port to other components which are part of the Top Level Composition
 - You connected the Det service port the compatible existing port in the SwcCyclicCoutner using the “Composition and Connection Editor”
 - You added the file „systemmod/ConnectionEditor.arxml“ to the previously configured System Importer.
This step ensures that the changes done with the Connection Editor will be preserved during a subsequent System Description import
 - You checked the „Implementation Selection“ in the RTE Editor. In our example no change is possible or required but the RTE Editor updates its configuration only after inspecting the assumed mapping manually
 - You used the “Calculate Service needs” wizard to apply the Os configuration required by the updated RTE configuration
- You rebuild the application and observed the changed behavior

Exercise 6: Com stack

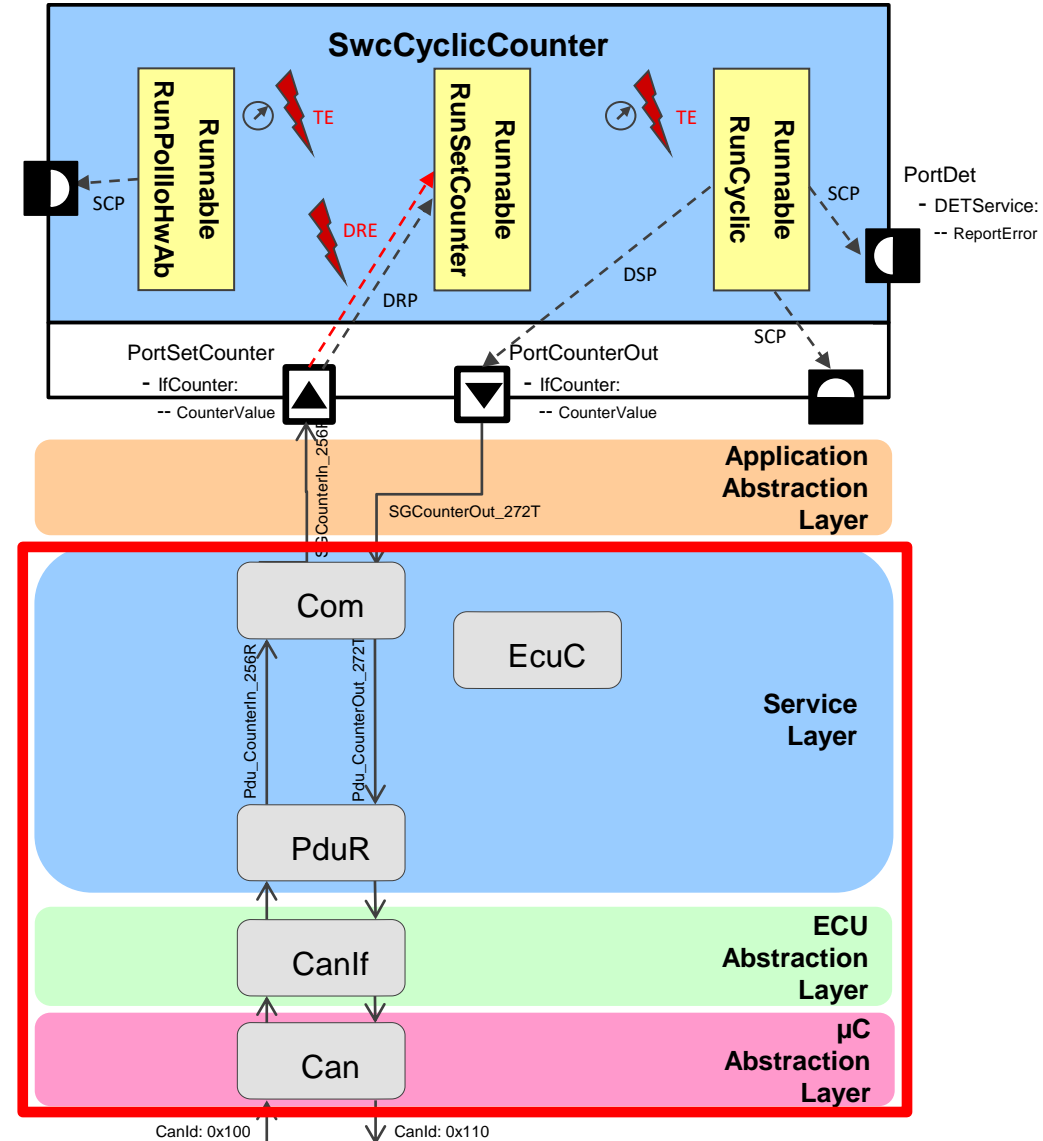


Elektrobit



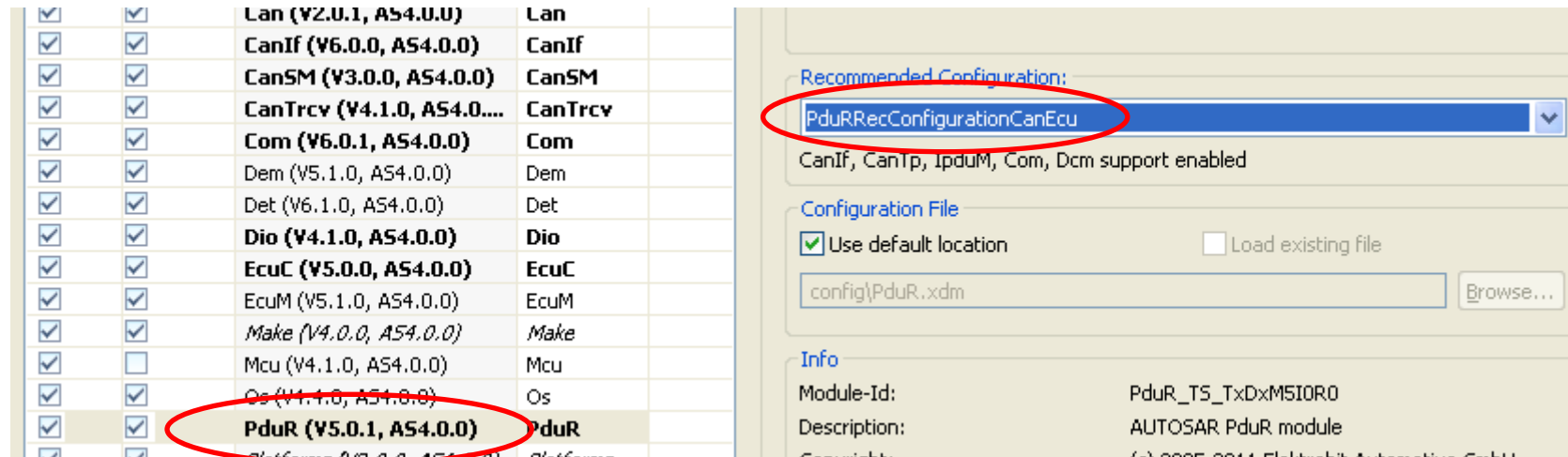
Com Stack Exercise – Objectives

- Add Communication stack modules to the project
- Execute wizards to derive Com stack configuration from the system model
- Complete Com stack configuration manually





- Add communication stack modules via "Module Configurations":
 - EcuC, Can, CanIf, Com (Default settings)
 - PduR (select recommended configuration (PduRRecConfigurationCanEcu))

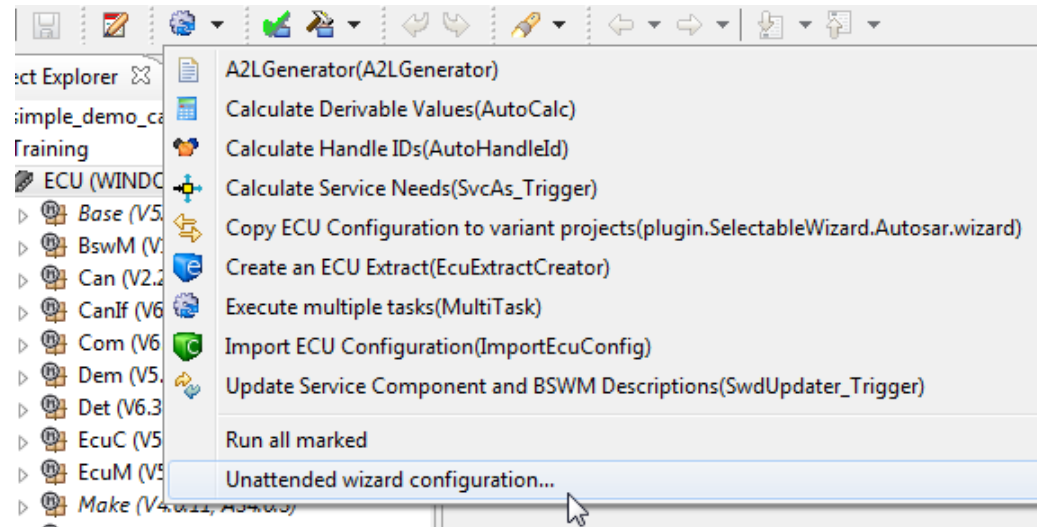


The screenshot displays the 'Module Configurations' dialog. On the left, a table lists modules with checkboxes for selection. The 'PduR (V5.0.1, AS4.0.0)' module is highlighted with a red circle. On the right, the 'Recommended Configuration' dropdown menu is also highlighted with a red circle and contains the text 'PduRRecConfigurationCanEcu'. Below this, the text 'CanIf, CanTp, IpduM, Com, Dcm support enabled' is visible. The 'Configuration File' section shows 'Use default location' checked and a text box containing 'config\PduR.xdm'. The 'Info' section at the bottom shows 'Module-Id: PduR_TS_TxDxMSIOR0' and 'Description: AUTOSAR PduR module'.

Module Name	Version	AS4.0.0	Module Name
Lan	(V2.0.1, AS4.0.0)		Lan
CanIf	(V6.0.0, AS4.0.0)		CanIf
CanSM	(V3.0.0, AS4.0.0)		CanSM
CanTrcv	(V4.1.0, AS4.0.0)		CanTrcv
Com	(V6.0.1, AS4.0.0)		Com
Dem	(V5.1.0, AS4.0.0)		Dem
Det	(V6.1.0, AS4.0.0)		Det
Dio	(V4.1.0, AS4.0.0)		Dio
EcuC	(V5.0.0, AS4.0.0)		EcuC
EcuM	(V5.1.0, AS4.0.0)		EcuM
Make	(V4.0.0, AS4.0.0)		Make
Mcu	(V4.1.0, AS4.0.0)		Mcu
Os	(V1.1.0, AS4.0.0)		Os
PduR	(V5.0.1, AS4.0.0)		PduR

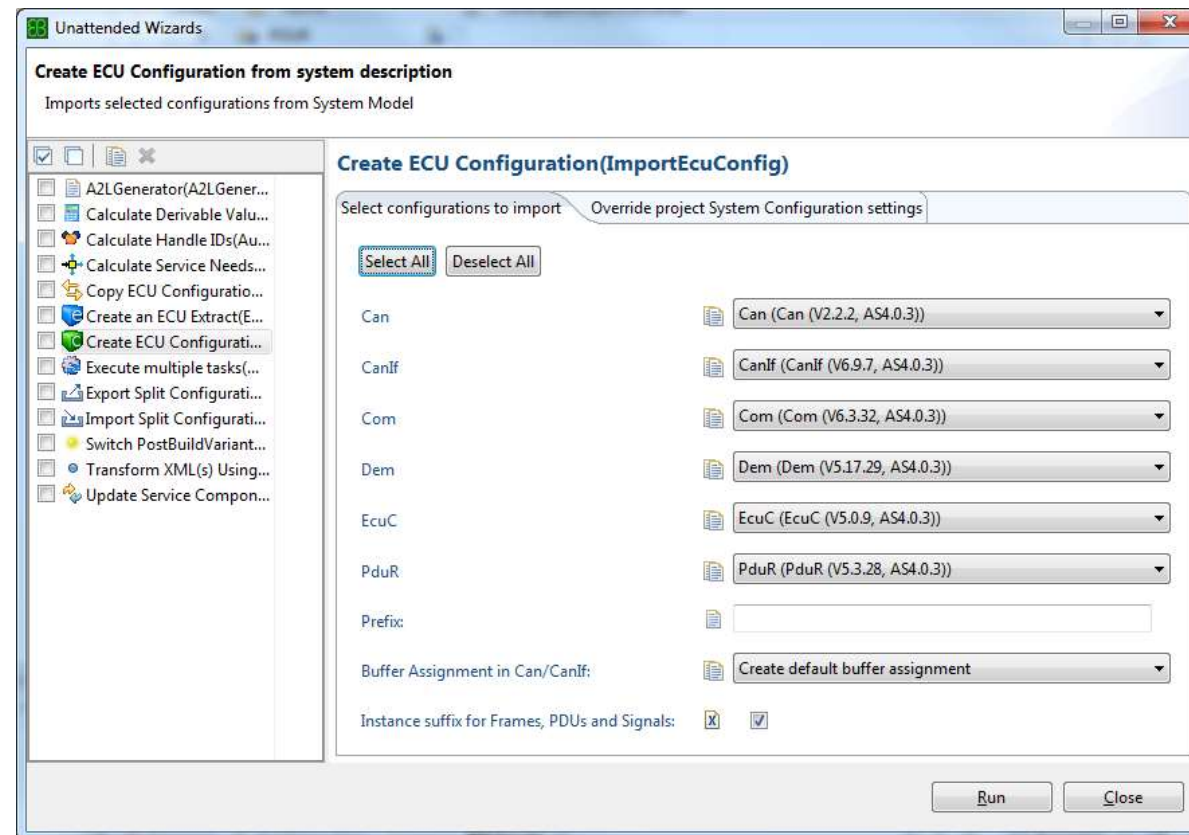


- PduR – tab PduRBSwModules:
 - Disable non-available modules: Dcm, CanTp, IpduM
- CanIf – tab CanIfUpperLayerConfig:
 - Delete non-available modules: CanNm, CanTp, CanTysn
- Configure the Unattended Wizard “Import ECU Configuration”



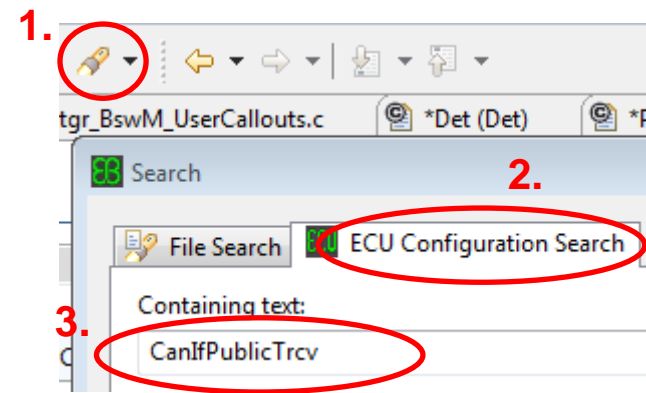


- Select ECU Configurations to import for all available modules:
 - „Select All“
- Run the wizard





- Run the "Calculate Handle Ids" wizard
- Search to find and de-select: CanIf → General →
 - CanIfPublicTrcvSupport
 - CanIfPublicTrcvWakeupSupport
 - CanWakeupSupport
 - CanHardwareCancellation



- Problem view should be without errors now
- Use Sidebar Editor "Edit System Signal Mappings" to inspect the signal connections



- Run „generate_swcd“
- Run System Description Importer again
- Run „Create an Ecu Extract“



Hint: In the unattended wizard configuration you can create your own multiple task wizard for combining repetitive tasks

- Start Rte editor, switch to tab „Event Mapping“
 - Map new Bsw events using the Auto-map feature:
- Close Rte editor



- Run "Calculate service needs" wizard
- Generate code
- No rebuild / debugging is required at this state because the application requires one further step: Starting the communication → see next step of the exercise

Com Stack - Accomplishments



- At this phase of the exercise you accomplished the following:
 - You added additional BSW modules to the project configuration. These modules were not part of the basic template configuration
 - You applied a „recommended“ configuration for the PduR at the time of adding this module to the project configuration
 - You configured major parts of the Com Stack configuration by executing the “Import ECU Configuration” Wizard
 - Note: This Wizard transforms the relevant information from the EB tresos internal System Model (which was imported before) to actual configuration parameters. These configuration parameters can be observed and distinguished by the yellow overlay Icon
 - You applied required manual configuration, mainly for removing unused features or support for BSW modules which are not used in the project
 - You executed the “Calculate Handle Ids” wizard to perform an automatic (re)calculation of the Handle Ids
 - Note: Prior to executing this wizard there are a lot of errors in the Problem View related to inconsistent or missing Handle Ids
 - You repeated the execution of the following wizards to update the Sytem Model / corresponding configuration
 - Run „generate_swcd“ → the SWCD of the added BSW modules are required for the System Description Importer
 - Run System Description Importer again → The actual System Description Importer needs to be executed
 - „Create an ECU Extract“ → Updating the EB tresos internal System Module with an updated ECU Extract
 - (Auto-)Map Bsw events in Rte editor → Mapping of any unmapped events to Os tasks
 - “Calculate Service needs” → Apply the required resulting Os configuration

Exercise 7: Mode Management



Elektrobit

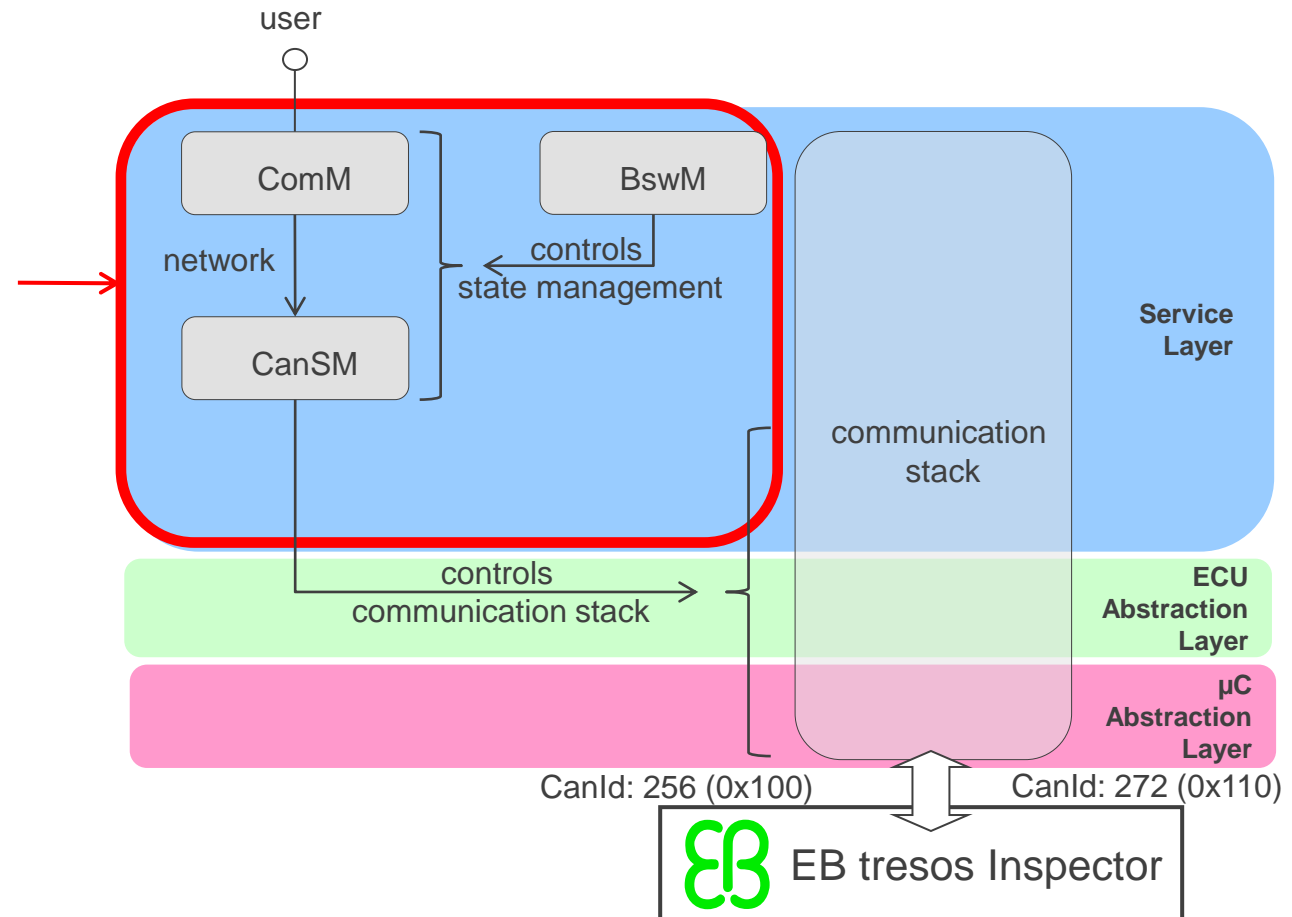




Mode management exercise

- Objectives

- add ComM, CanSM
- update ComM
- update BswM






- Add the following modules to the project:
 - CanSM
 - ComM

- Open configuration settings of „Create Ecu Configuration“ wizard
 - Click „Select All“ to enable all modules, including the newly added ones
 - Run the wizard


- In ComM → General
 - Disable Dcm Usage

- In ComM → ComMConfigSet/ComMUser
 - Add a new ComM User in the configuration (keep default names)
 - Reference this user in „Users Per Channel“

Network Channel

Name  CanNetwork

General **User Per Channel**

 User Per Channel*

Index	Name	User Reference
0	ComMUse...	/ComM/ComM/ComMConfigSet/ComMUser_0



- Reconfigure BswM by importing an existing prepared configuration set to the BswM configuration
 - Copy all files from „ C:\EB_Training\AddStuff\BswMforComStack“ to the project
C:\EB_Training\tresos_Wincore\workspace\Training
 - Reload Project
 - Run Wizard SplitableImp_ComRule

- Fix the the errors related to the follwoing parameters by using the autocalulation feature:
 - BswMMaxNumActionList
 - BswMMaxNumRules



- Run „generate_swcd“
- Run System Description Importer again
- Run „Create an ECU Extract“



Hint (again): In the unattended wizard configuration you can create your own multiple task wizard for combining repetitive tasks

- Check the mapping of Bsw events in the Rte editor. There should be nothing new at this point
- Run „Calculate service needs“ wizard
- Generate project.



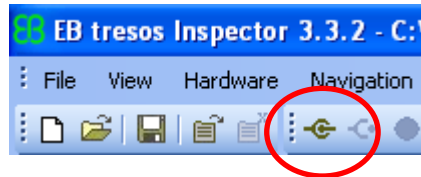
- Switch to eclipse CDT
 - Build project (according to previous exercises) (clean, make j)
 - Execute/debug application, e.g. set a breakpoint in Rte.c at
Rte_Write_SwcCyclicCounter_PortCounterOut_CounterValue



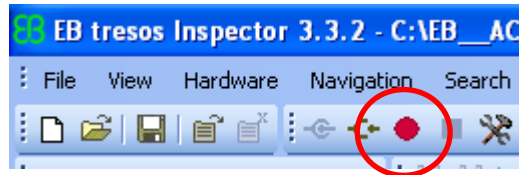
Optional – Only if you have a tresos Inspector license

- Start EB tresos inspector

- Open project C:\EB_Training\tresos_Inspector\Training.ipf
- Connect to hardware



- Start recording



- Switch to eclipse CDT

- Build project (according to previous exercises) (clean, make j)
- execute

- The counter value is also measured with inspector via virtual CAN frames

- Continue



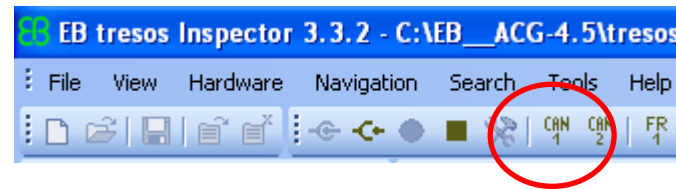
Optional – Use EB tresos Inspector

- The counter value is also transmitted via virtual CAN frames can be visualized with the tool EB tresos Inspector
 - The next 2 slides explain how to use EB tresos Inspector to observe the received CAN frame
 - **Continue** if you have a valid EB tresos Inspector license
 - Otherwise **skip the next 2 slides** and just watch the exercise demonstration / the recorded exercise video

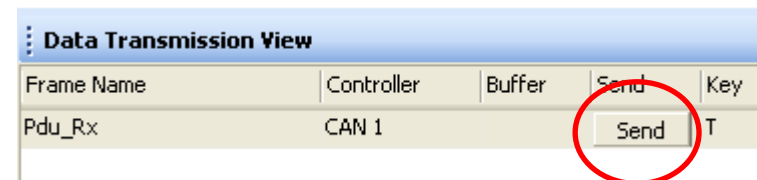


Optional – Only if you have a tresos Inspector license

- To be able to send CAN frames from inspector, the transmission must be enabled:



- Now a transmission could be issued by pressing the Send button.
- The CounterIn value is now transmitted over virtual CAN bus and sets the CounterValue to the transmitted value (0 by default)



Mode management - Accomplishments



- At this phase of the exercise you accomplished the following:
 - You added additional modules for mode management which were not part of the project configuration
 - You re-run the “Create Ecu Configuration” Wizard to derive the configuration from the EB tresos internal System model also to these added modules
 - You updated the BswM configuration by using an importer
 - Note: The reason for providing a prepared configuration is that the manual configuration would consume too much time. Please observe the changes applied by the updated configuration
 - You performed required manual configuration, mainly a minimal configuration of the ComM including 1 user and link it to the communication channel
 - You repeated the execution of the following wizards to update the Sytem Model / corresponding configuration
 - Run „generate_swcd“ → the SWCD of the added BSW modules are required for the System Description Importer
 - Run System Description Importer again → The actual System Description Importer needs to be executed
 - „Create an ECU Extract“ → Updating the EB tresos internal System Module with an updated ECU Extract
 - (Auto-)Map Bsw events in Rte editor → Mapping of any unmapped events to Os tasks
 - “Calculate Service needs” → Apply the required resulting Os configuration
 - You rebuild the application and observed the behavior by using EB tresos Inspector

Exercise 8: Memory Stack



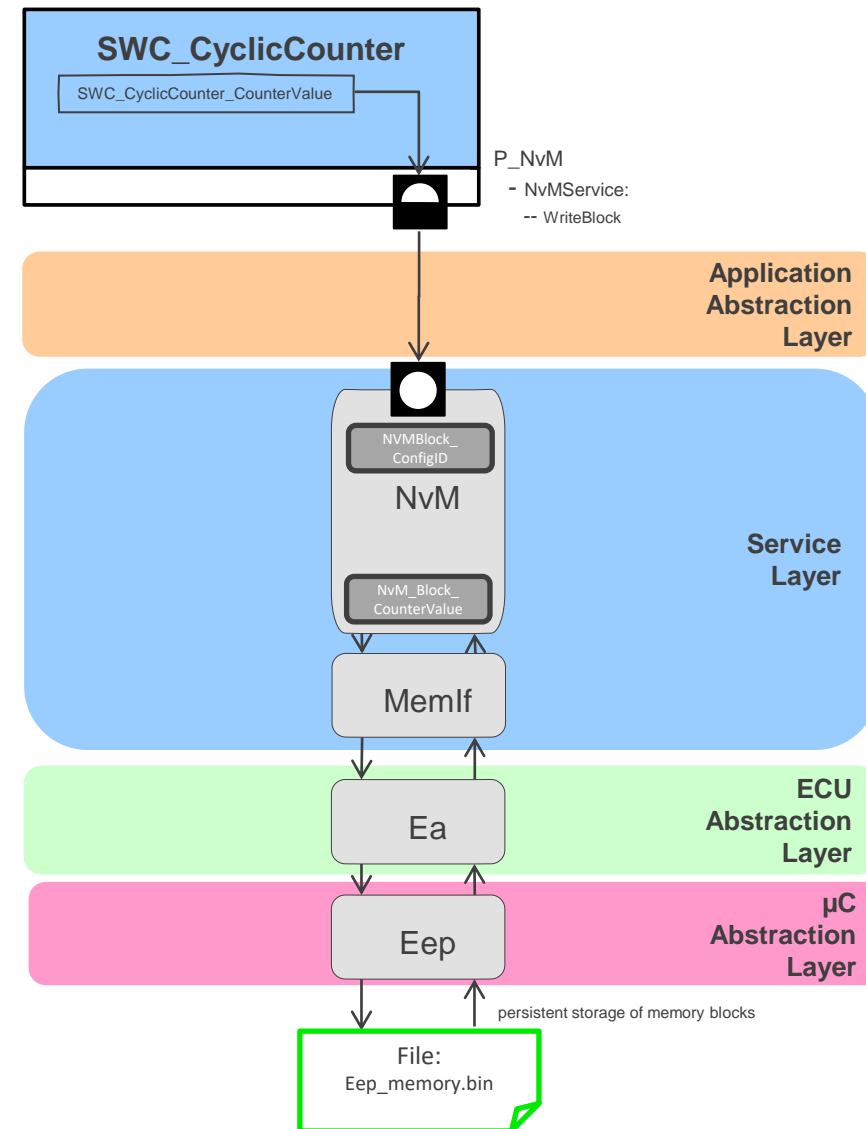
Elektrobit



Memory Stack exercise

Objectives

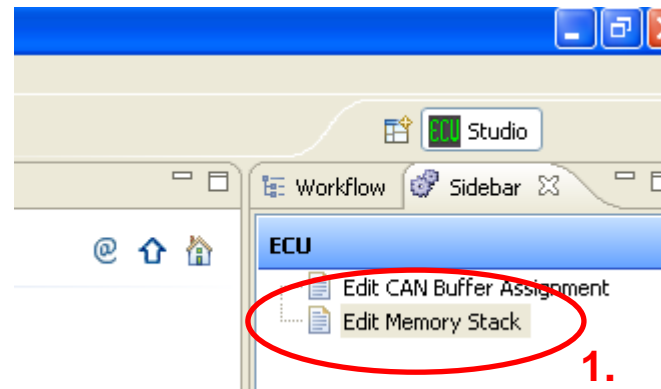
- Add memory stack modules
- Work with memstack editor
- Complete configuration





- Add memory stack modules
 - NvM
 - Crc
 - MemIf
 - Ea
 - Eep

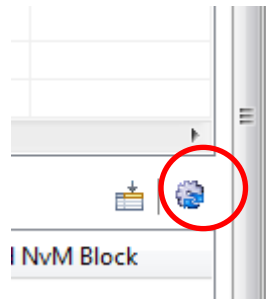
- Open the “Memory stack editor” from the Sidebar, Section “ECU”





Edit in Memory Stack Editor (Sidebar)

- Add block for application:
 - Name: NvM_Block_Counter
 - RAM Block Address: &SwcCyclicCounterCounterValue
 - ROM Block Address: &SwcCyclicCounterCounterValueDefault
 - ReadAll: true
- Start Autoconfiguration and close memory stack editor



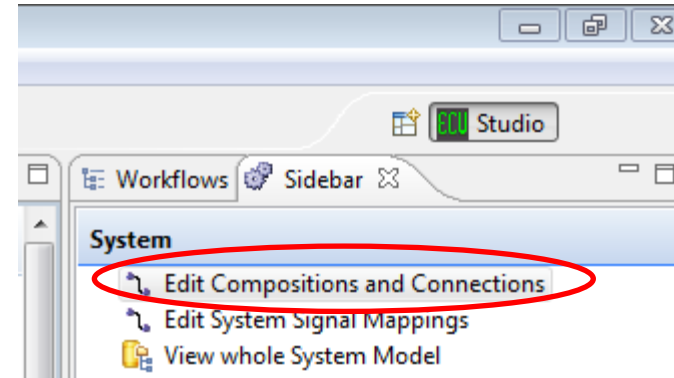
Edit in NvM Generic Editor (Project Explorer)

- Manual configuration:
 - Tab General:
 - Deselect „Enable BSWM Multi Block Job Status Information“
 - Enable „Multiblock Request Call-back Function“
 - Enable “Size of Queue for Immediate Requests”
 - Set “Enable RTE Usage” to “true”
 - Tab Block Configuration:
 - Enable „Enable Rte Service Port“ for NvM_Block_Counter
 - Tab User Header
 - Add „SwcCyclicCounter.h“
- Update System Model
 - Run „generate_swcd“
 - Run System Description Importer again
 - Run „Create an Ecu Extract“

Hint: you can create your own multiple task wizards for repetitive tasks.



- Open Connection editor
 - Add a prototype for the NvM
 - Connect SwcCyclicCounterInstance → PortNvM with NvM
 - Close connection editor
- Run „Create ECU extract“ wizard
- Open Rte editor
 - „Event Mapping“: map Bsw events
 - Close Rte editor
- Run „Calculate service needs“ wizard
- Generate project
- Switch to eclipse CDT, build project, terminate and restart:
(counter value is now stored)





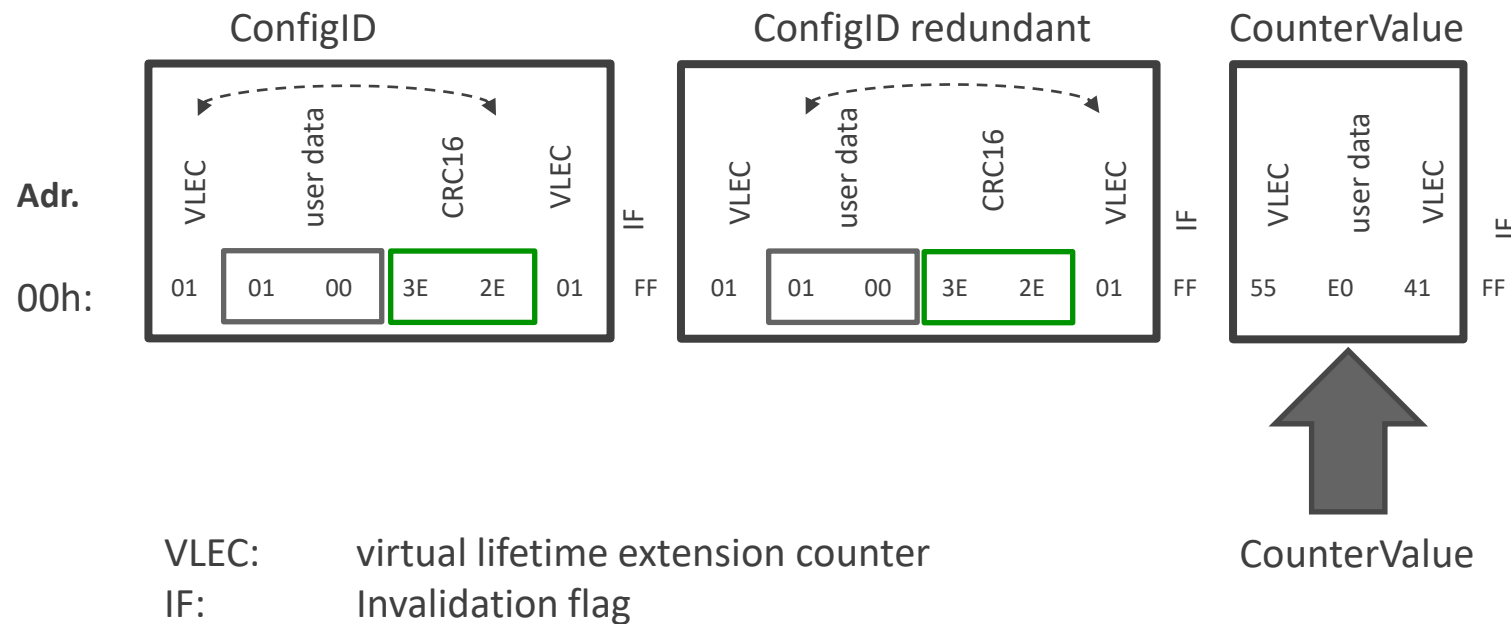
File content (binary file!)

- If you execute from eclipse CDT without debugging session:

C:\EB_Training\tresos_Wincore\workspace\Training\output\bin\Eep_memory.bin

- If you started debugging session:

C:\EB_Training\EclipseProjects\workspace\Training\Eep_memory.bin



Memory - Accomplishments



- At this phase of the exercise you accomplished the following:
 - You added additional modules for memory stack which were not part of the project configuration
 - You used the Memory Stack Editor (in the Sidebar) to configure an NvM Block that can be used by the application
 - You used the generic editor of the NvM to configure additional parameters manually
 - You added the NvM instance to the Top Level composition and connected the NvM Port to the application using the connection editor
 - You repeated the execution of the following wizards to update the Sytem Model / corresponding configuration
 - Run „generate_swcd“ → the SWCD of the added BSW modules are required for the System Description Importer
 - Run System Description Importer again → The actual System Description Importer needs to be executed
 - „Create an ECU Extract“ → Updating the EB tresos internal System Module with an updated ECU Extract
 - (Auto-)Map Bsw events in Rte editor → Mapping of any unmapped events to Os tasks
 - “Calculate Service needs” → Apply the required resulting Os configuration
 - You rebuild the application and observed the changed behavior

Exercise 9: IO Hardware Abstraction example

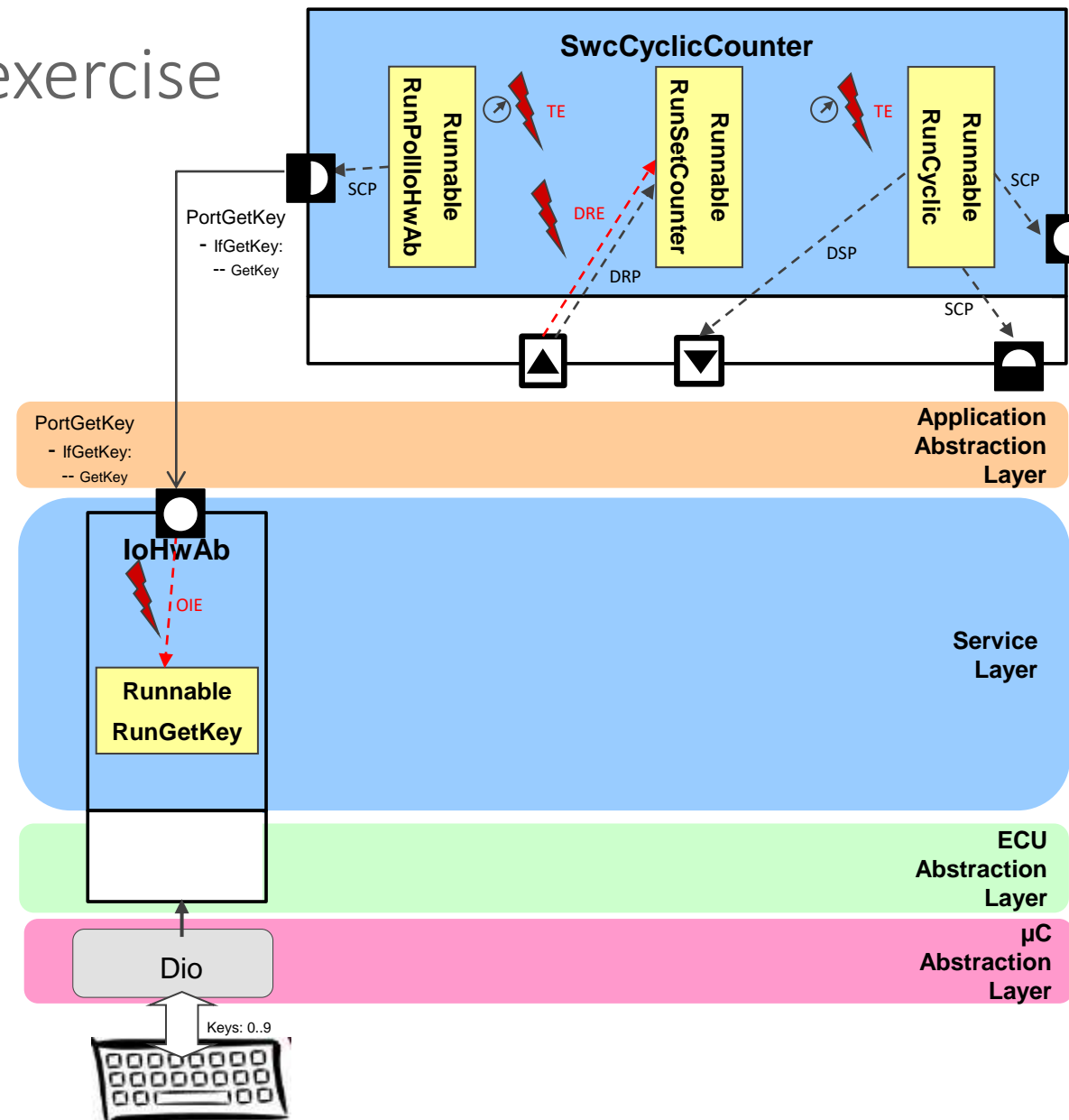


Elektrobit



IO Hardware Abstraction exercise

- Objectives
 - Work on an example of an IO Hw Abstraction





- In EB tresos Studio:
 - Add Dio module in EB tresos Studio module configuration
 - Generate Code

- In C:\EB_Training\tresos_Wincore\workspace\Training\source\application\IoHwAb.c
- Enable Dio include:
`#include "Dio.h"`

- Enable call to Dio_ReadChannel, remove
`#if 0`
`#endif`

- Build project in eclipse CDT
 - run program, press keys (0..9) and check Counter value

IoHwAb - Accomplishments



- At this phase of the exercise you accomplished the following:
 - You added the Dio module which is used in this training Exercise for simulating I/O access
 - You altered the actual sample application of the IoHwAb to the project to make use of the added Dio module
 - You rebuild the application and observed the changed behavior

Get in touch!



Elektrobit

sales@elektrobit.com
www.elektrobit.com

