Exercise 1 – Software Components

Software Components and RTE

Get familiar with DaVinci Developer, design Software Components, and connect them via ports. Create the Runnables that will contain your code. Have a first look at DaVinci Configurator Pro to configure the OS.

Generate the BSW configuration and fill the runnable skeletons with your code.

The target of this first exercise is to generate the RTE and test the result via a breakpoint in Visual Studio.

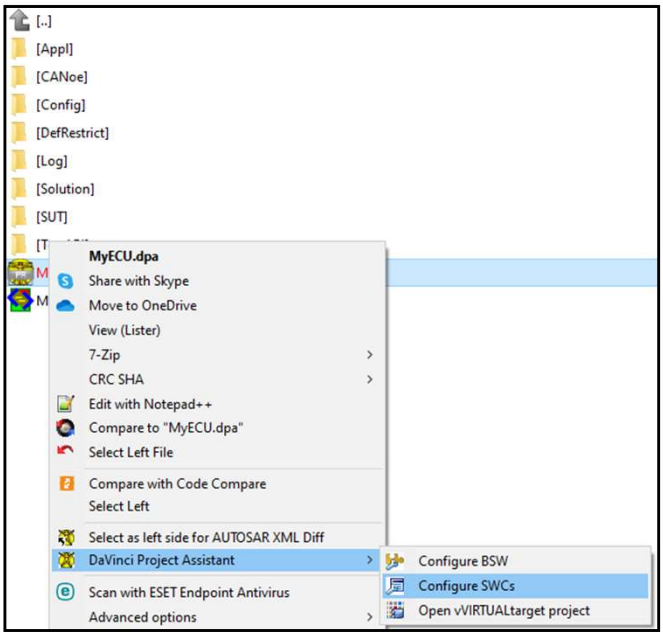
Exercise 1 – Software Components

Exercise steps

1. Design Software Components, Ports, Data Elements, Connections
2. Create Runnables and Tasks
3. Configure BSW
4. Generate BSW, implement SWC functionality and test it

Exercise 1 – Software Components
Part 1/4: Software Components

Open **DaVinci Developer**
Right click on _E1_SoftwareComponents\ MyECU.dpa,
select DaVinci Project Assistant->Configure SWCs



Exercise 1 – Software Components
Part 1/4: Software Components

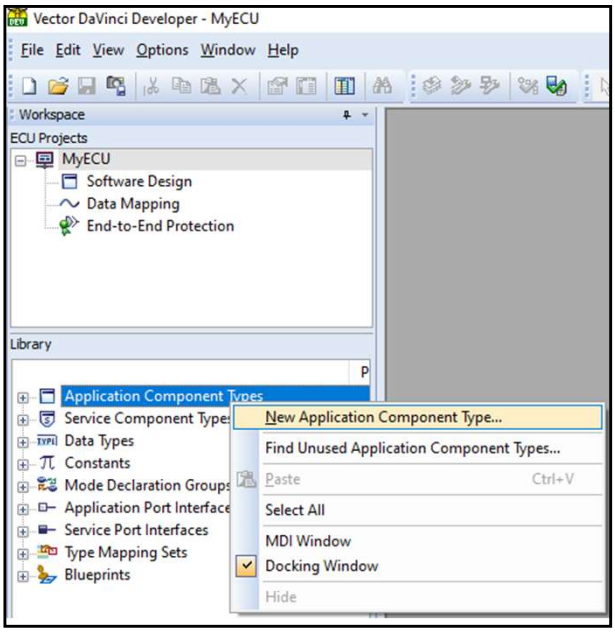
Create following Application Component Types:

- Name: **CtCoApplication**
- Type: Composition

- Name: **CtApMySwc**
- Type: Application
- Support Multiple Instantiation: OFF

- Name: **CtSaDoor**
- Type: SensorActuator
- Support Multiple Instantiation: ON

- Name: **CtSaInteriorLight**
- Type: SensorActuator
- Support Multiple Instantiation: ON



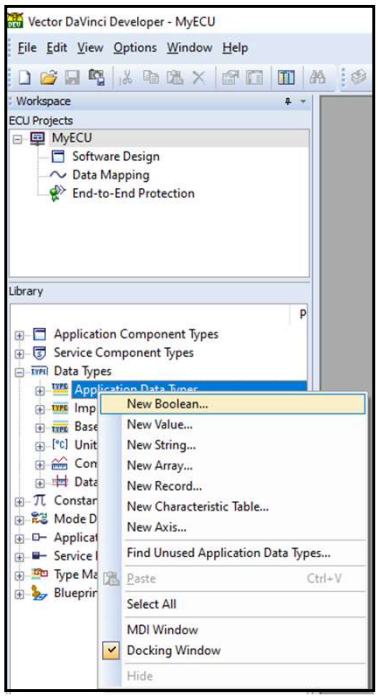
Component Types which may be instantiated more than once in an ECU should have **Support Multiple Instantiation** set to **On**.

The Component Type **CtApMySwc** will only be instantiated once in the ECU because it contains the application algorithm.
Other component types may exist more than once (i.e., four doors, two interior lights).

Exercise 1 – Software Components
Part 1/4: Software Components

Create an **Application Data Type** (in the Developer Library;
(see right)

- **AdtDoorState** as New Boolean Type
- **AdtLightState** as New Boolean Type

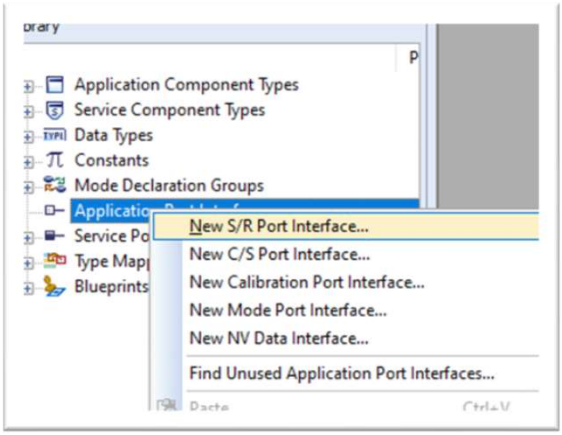


Exercise 1 – Software Components
Part 1/4: Software Components

Create following Application S/R Port Interfaces:

- Name: **PiDoorState**
- Data Element Name: **DeDoorState**
- Data Element Type: **AdtDoorState**

- Name: **PiLightState**
- Data Element Name: **DeLightState**
- Data Element Type: **AdtLightState**



The port prototypes have to transport information about the door states and light states. Define one data element of the appropriate type for each port interface before port prototypes are instantiated.

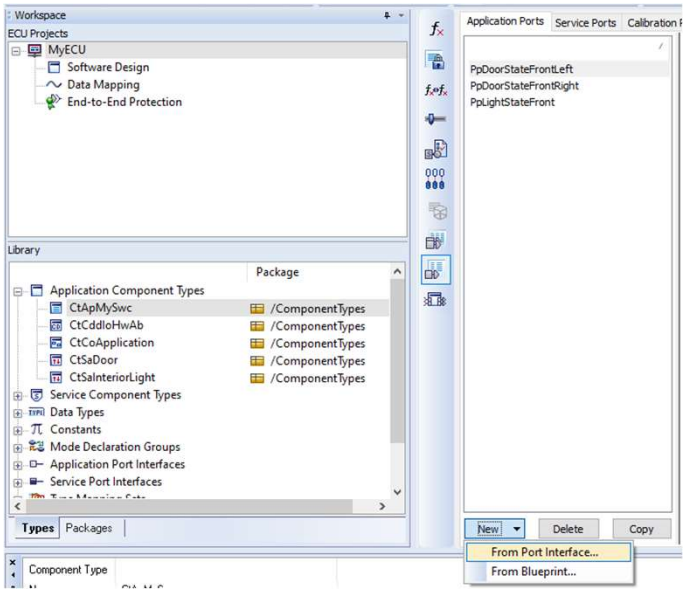
Exercise 1 – Software Components
Part 1/4: Software Components

Create following Port Prototypes on **CtApMySwc**:

› Receiver port **PpDoorStateFrontLeft**
Port interface: **PiDoorState**

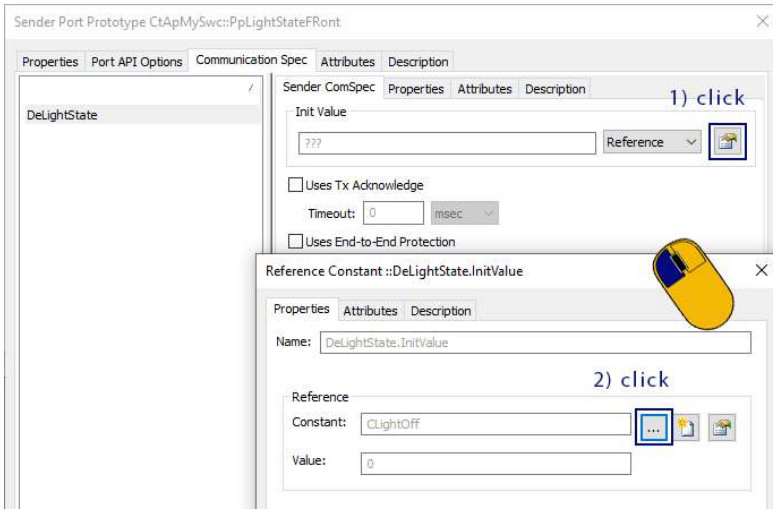
› Receiver port **PpDoorStateFrontRight**
Port interface: **PiDoorState**

› Sender port **PpLightStateFront**
Port interface: **PiLightState**



Exercise 1 – Software Components
Part 1/4: Software Components

For created ports select following Init values:
› **PpDoorStateFrontLeft**, Init value: **CDoorClosed**
› **PpDoorStateFrontRight**, Init value: **CDoorClosed**
› **PpLightStateFront**, Init value: **CLightOff**



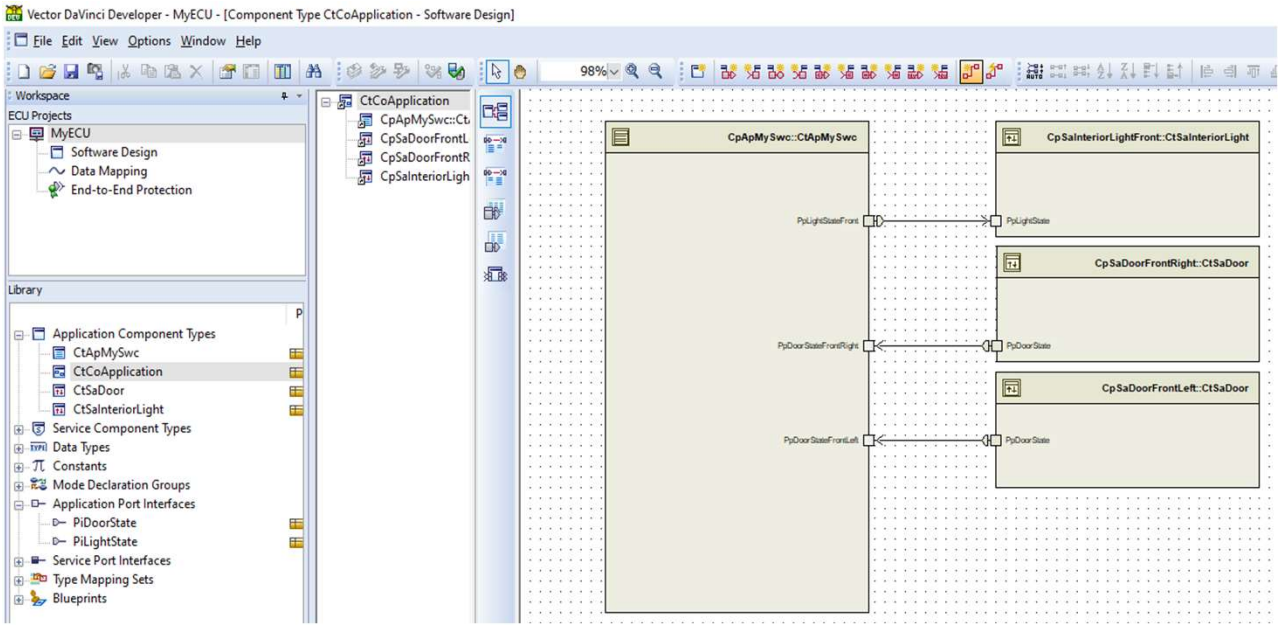
Assignment of Init Value for data elements:
- Select Application Port and data element for which Init value should be set
- As Init Value type select Reference
- Choose properties 1) on image above
- Select existing constant 2) on image above

Exercise 1 – Software Components
Part 1/4: Software Components

Create additional Port Prototypes:

› Sender port **PpDoorState** on **CtSaDoor**
Port interface: **PiDoorState**
Init value: **CDoorClosed**

› Receiver port **PpLightState** on **CtSaInteriorLight**
Port interface: **PiLightState**
Init value: **CLightOff**



Open the **CtCoApplication** by double click on it and instantiate all Component Prototypes inside it (drag and drop from Library to Software Design Sheet).
Draw the Connectors between Component Prototypes like on picture above (use Draw Connector option, highlighted on toolbar above).

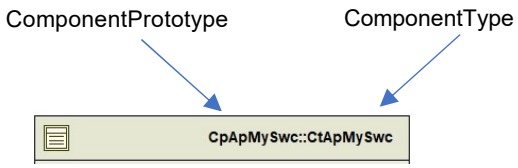
Exercise 1 – Software Components
Part 1/4: Software Components

Please note the complete software design is contained in the composition **CtCoApplication**. This composition must contain the atomic software components you want to use in your software design.

As a final step, you must instantiate the **CtCoApplication** in the Software Design in DaVinci Developer.

The Component Type **CtSaDoor** is used twice, two different instances (Component Prototypes) of the same **CtSaDoor** exist (**CpSaDoorFrontLeft** and **CpSaDoorFrontRitght**).

By connecting the Port Prototypes (**PpDoorState**) to different Port Prototypes of other Component Prototypes (**PpDoorStateFrontLeft, PpDoorStateFrontRight**), each instance has the same behavior, but with different input and output.

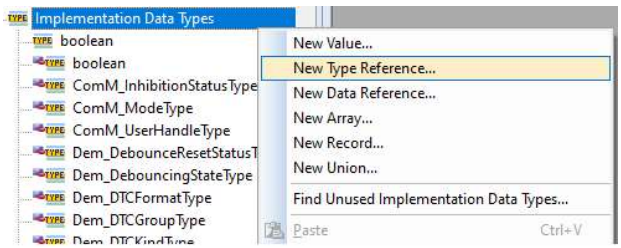


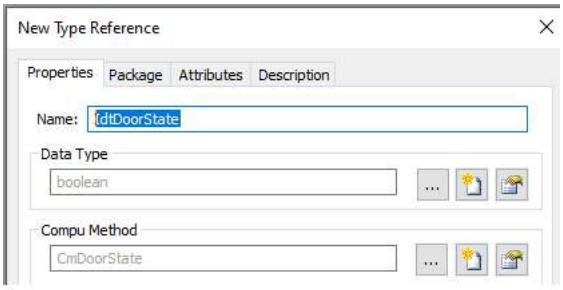
Exercise 1 – Software Components
Part 1/4: Software Components

Create the following Implementation Data Type References:

- Name: **IdtDoorState**
- Data Type: **boolean** from /DataTypes/PlatformTypes
- Compu Method: **CmDoorState**

- Name: **IdtLightState**
- Data Type: **boolean** from /DataTypes/PlatformTypes
- Compu Method **CmLightState**



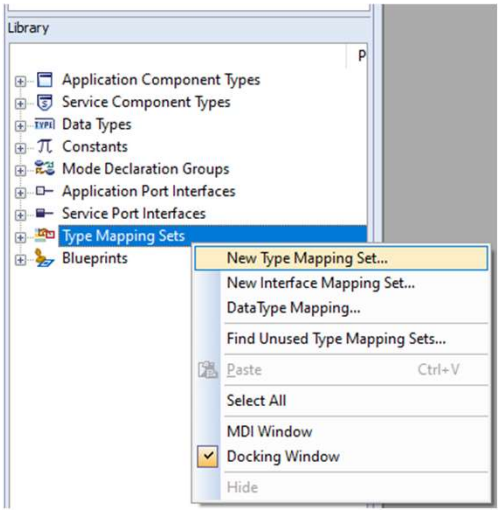


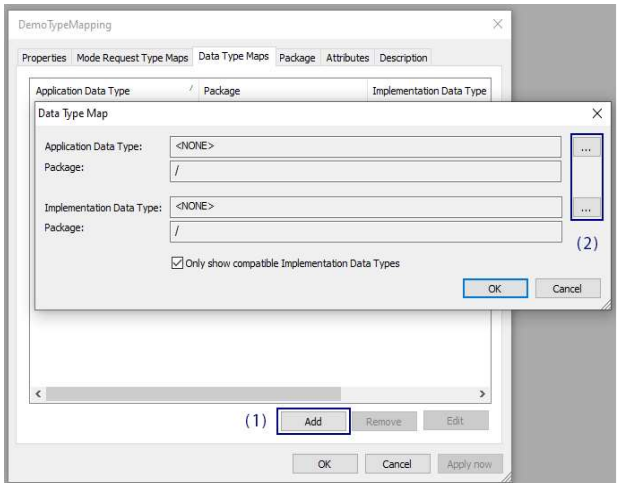
HINT: The Compu Method can be used to generate #defines into the SW component header file **Rte_<SwCTypeName>_Type.h.**

In our case these defines are:
CMLIGHTSTAE_LIGHTON/LIGHTOFF
and
CMDOORSTATE_DOOROPEN/CLOSED

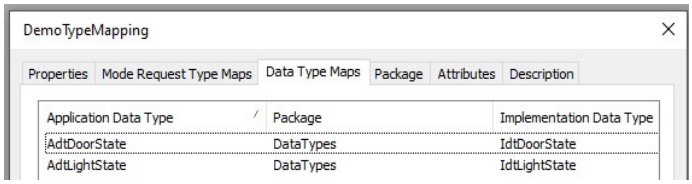
Exercise 1 – Software Components
Part 1/4: Software Components

Create a Type Mapping Set with the name
DemoTypeMapping





Creation of a Data Type Map:
(1) click "Add"
(2) select Application Data Type and Implementation Data Type mapping



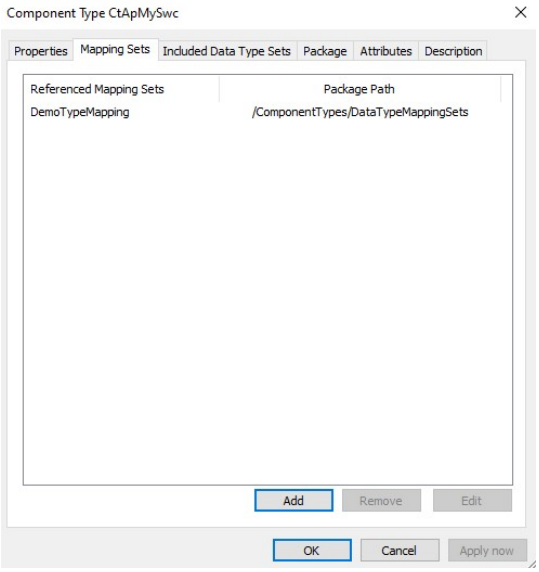
Result: the application data types are mapped to implementation data types

Define a **Data Type Map** for DemoTypeMapping

Assign an Implementation Data Type to each Application Data Type
- **AdtDoorState → IdtDoorState**
- **AdtLightState → IdtLightState**

Exercise 1 – Software Components
Part 1/4: Software Components

Now it is time to assign the Type Mapping Set to all atomic SWCs in your design.

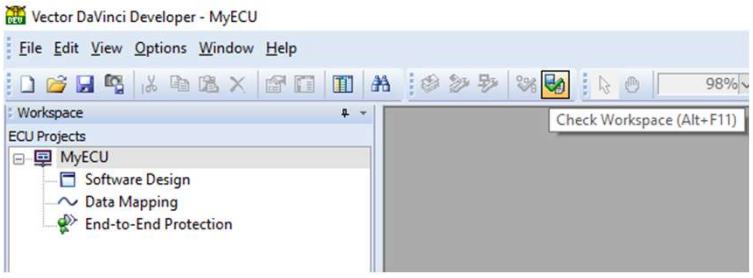


For Each Atomic SWC Type
- Open the Component Type Properties (see upper right figure)
- Assign the Type Mapping Set to the SWC (see right figure)

Hint: You have 3 atomic SWC Types (**CtApMySwc**, **CtSaDoor** and **CtSaInteriorLight**).

After this, you can perform validation of your workspace. This is in the main Developer toolbar (see image bellow). If **no** type mapping set is selected, DaVinci Developer reports in its message window: 40317 Missing data type mapping detected

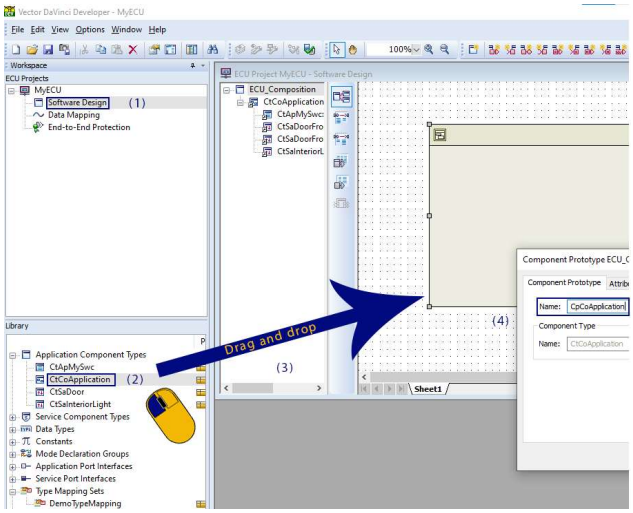
You should **not** see this specific message (error 40317) if you have successfully assigned the data type maps to your SWC Types!
Other warnings are OK at this point.



Exercise 1 – Software Components
Part 1/4: Software Components

Instantiate the Composition **CtCoApplication** in the Software Design view.

1. Double click on MyECU->Software Design
2. Select **CtApplication** from Library
3. Drag and drop **CtCoApplication** from Library to Software Design Sheet
4. Right click on the newly created **CtCoApplication**, select "Properties" and rename Component Prototype to **"CpCoApplication"**



When **Component Prototype** are created, they keep same name as **Component Type**, so you have to rename them manually (described in step 4 above).
For Component Type names prefix **Ct** is used, but for Component Prototype names we use prefix **Cp**.

Additionally, if Component Type has multiple Component Prototypes, Component Prototypes must have names that differentiate them from each other
(i.e., **CtSaDoor** -> **CpSaDoorFrontLeft**, **CtSaDoor_1** -> **CpSaDoorFrontRight**, **CtApMySwc** -> **CpApMySwc**, **CtSaInteriorLight** -> **CpSaInteriorLightFront**).

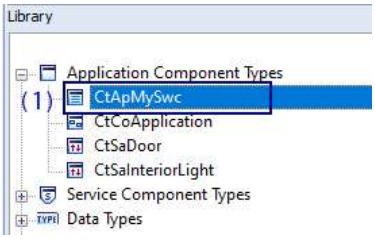
Exercise 1 – Software Components
Part 2/4: Runnables and Tasks

The next step defines the internal behavior for the atomic SWC Types

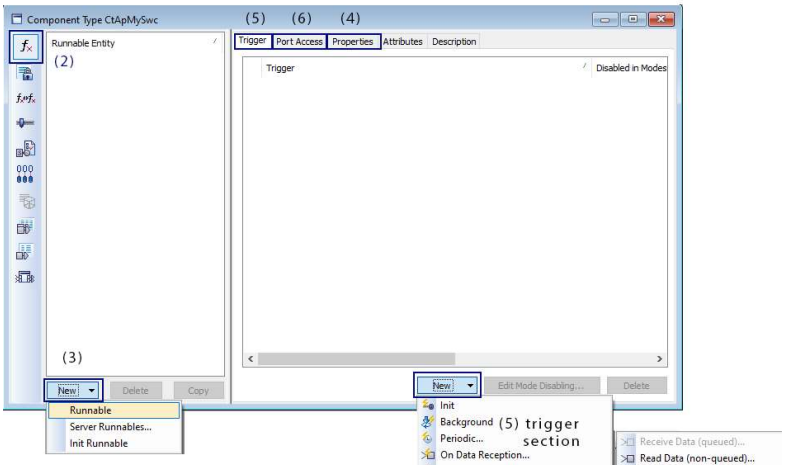
Each atomic SWC Type contains at least one "Runnable"

- Each Runnable usually has a "trigger"
- Each Runnable has “port access" in order to receive and send data, or call an operation
- The procedure for defining Runnables, their triggers, port accesses and names is illustrated on next slides

HINT: The RTE controls the execution of Runnables. The trigger condition decides when a Runnable will run (cyclically, on data reception, etc.).
Port Access is used to allow runnables to access data elements or receiver or sender ports



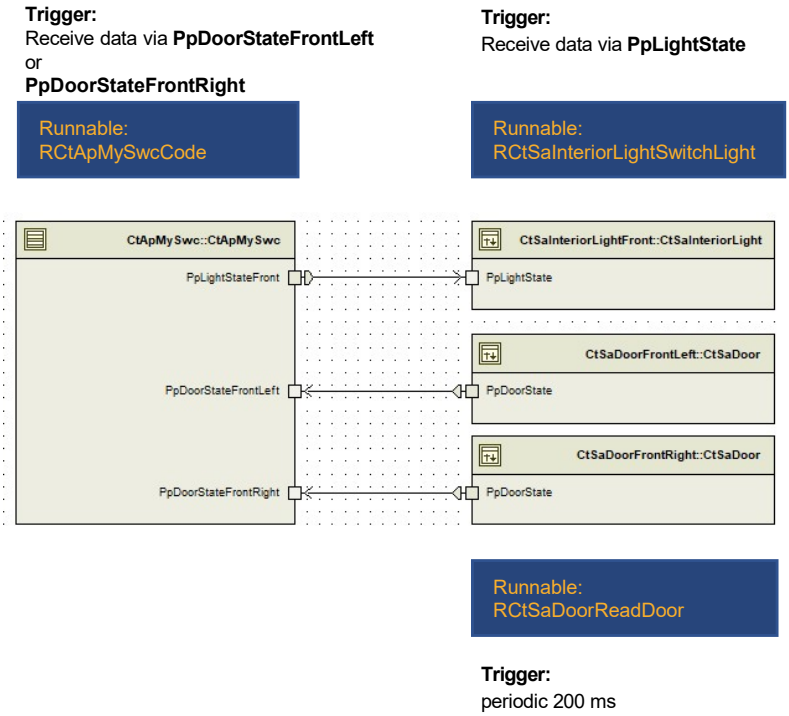
Double click the SWC Type (2) to access Component Type view



Select 'f(x)' icon to access runnable dialog (2). Create a New **Runnable** (3) and give it a name, (4) select Trigger in tab (5),select the Port Access in tab (6)

Exercise 1 – Software Components
Part 2/4: Runnables and Tasks

- Define Runnables with the **Triggers** given in the figure.
- Define the correct **Port Access** for each Runnable.
- Check workspace, save and close Developer.



Add Port Access for following Runnables:

CtSaDoor.RCtSaDoorReadDoor
- **write explicit** for **PpDoorState.DeDoorState**

CtSaInteriorLight.RCtSaInteriorLightSwitchLight
- **read explicit by argument** for **PpLightState.DeLightState**

CtApMySwc. RCtApMySwcCode
- **read implicit** for **PpDoorStateFrontLeft.DeDoorState**
- **read explicit by argument** for **PpDoorStateFrontRight.DeDoorState**
- **write explicit** for **PpLightStateFront.DeLightState**

Exercise 1 – Software Components
Part 2/4: Runnables and Tasks

OS Tasks are required to execute Runnables of our SWCs

In our design we have some Sensor/Actuator SWCs and control application that is independent of sensors and actuators.

We will map all our runnables to two OS tasks:

- **My_Task**, all Runnables from **CtApMySwc** are mapped to this task
- **IO_Task**, all Runnables from **CtSaDoor** and **CtSaInteriorLight** are mapped to this task

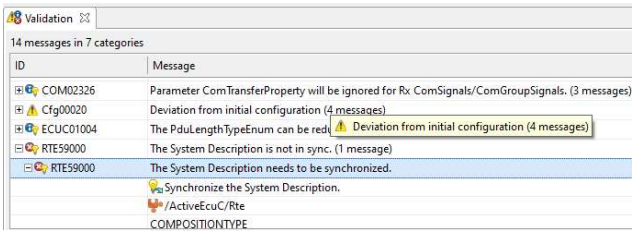
HINT: This slide summarizes only what tasks we need and for what we use them. See next slide for "how to" instructions.

Runnable **RCtSaDoorReadDoor** will be mapped twice, one time for each Component Prototype.

Exercise 1 – Software Components
Part 2/4: Runnables and Tasks

Open **DaVinci Configurator**
Right click on _E1_SoftwareComponents\ MyECU.dpa,
select DaVinci Project Assistant->Configure BSW

After opening Configurator, workspace will be Validated automatically. When validation finishes double-click on solving action, marked with light bulb "Synchronize the System Description".

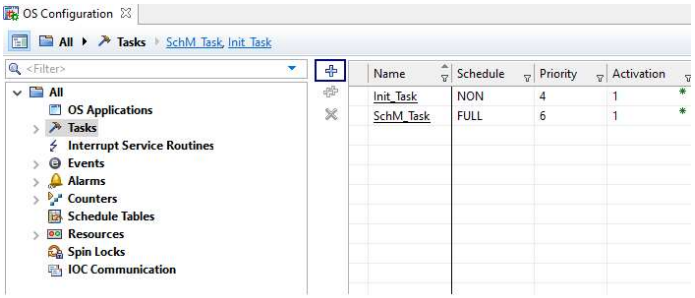


After this is done following RTE error will appear:
RTE 01056 Unmapped runnable entity (4 messages)

To fix this we have to map runnables to Taks
Go to **Runtime System**
Select **OS Configuration → Tasks**

There are already some OS tasks defined:
› Init_Task is an AUTOSTART task, and it is responsible for initialization of the Basic Software
› SchM_Task is responsible for the periodic execution of the BSW modules main functions

Don't make any changes to those tasks.



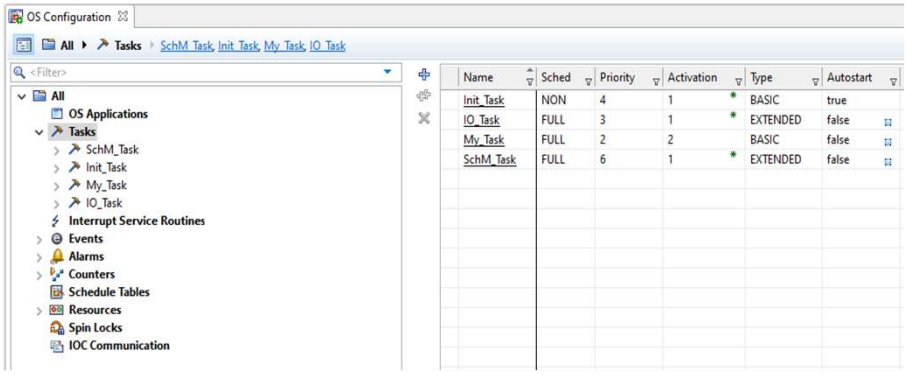
The OS Configuration dialog in Configurator Pro. Create new task by clicking the '+' icon.

Exercise 1 – Software Components
Part 2/4: Runnables and Tasks

Create following tasks:

- Name: **My_Task**
- Schedule: **FULL**
- Priority: **2**
- Activation: **2**
- Task Type: **Basic**

- Name: **IO_Task**
- Schedule: **FULL**
- Priority: **3**
- Activation: **1**
- Task Type: **Extended**



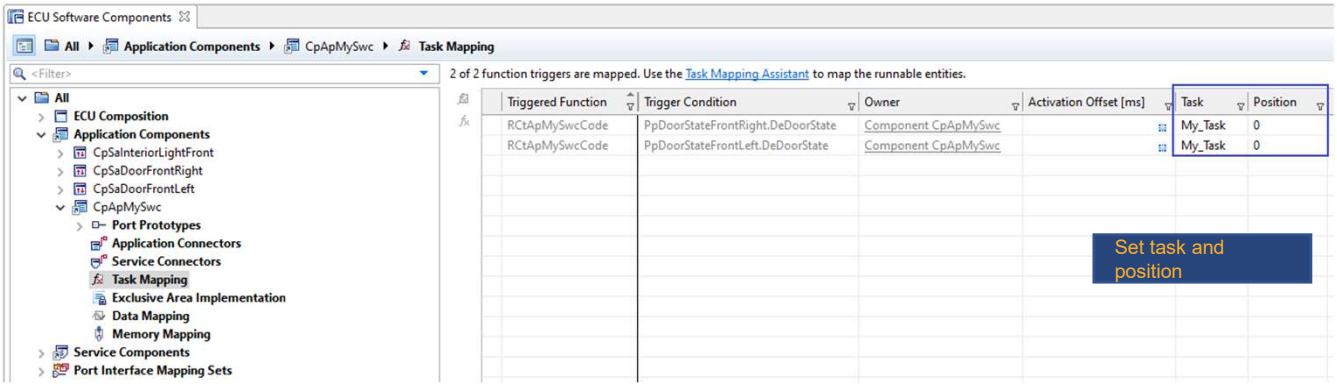
Resulting task configuration shown here

Exercise 1 – Software Components
Part 2/4: Runnables and Tasks

Select **ECU Software Components** (see right)
You will see all atomic SWC Prototypes in your design.

Task mapping for **CpApMySwc** :

› **RCtApMySwcCode** → **My_Task**
› Position: 0, for both triggers



Define the Task mapping for each SWC, same procedure as for **CpApMySwc**.

Task mapping for **CpSaDoorFrontLeft:**
› **RCtSaDoorReadDoor** → **IO_Task**
› Position: 0

Task mapping for **CpSaDoorFrontRight:**
› **RCtSaDoorReadDoor** → **IO_Task**
› Position: 1

Task mapping for **CpSaFrontInteriorLigh:**
› **RCtSaInteriorLightSwitchLight** → **IO_Task**
› Position: 2

Runnable **RCtSaDoorReadDoor** has to be mapped twice, one time for each Component Prototype.

Exercise 1 – Software Components
Part 3/4: Configure AUTOSAR OS

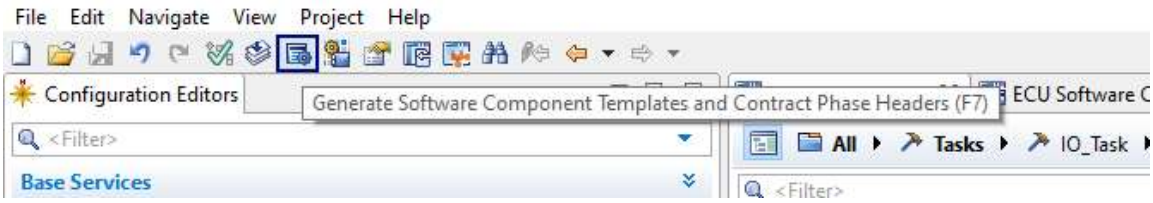
Open **OS Configuration**, the RTE has already added all required elements resulting from the SWC Design: Alarms, Events, etc.

Save the project

Generate BSW (press F9)
Click "OK" to close the `Build VTT project' Dialog
Close the Generation Dialog

Press F7 to generate the SWC Templates
(see next slide for more details)

(1) Please remember the SWC Templates also have to be updated. In the current version of the Configurator tool, this must be done using a separate icon next to the generate BSW icon.



Exercise 1 – Software Components
Part 4/4: Generate code, program Runnables and test it

Open Visual Studio Project by double-clicking on file
_E1_SoftwareComponents\Solutions\MyECU.sln

Implement SWC functionality inside generated implementation templates (**CtApMySWC.c**, **CtSaDoor.c**, **CtsaFrontInteriorLight.c**)

In **CtsaDoor.c**

- simulate the state of the front doors, hard-code door state to open or close for testing purposes, in this exercise we are not connected to door sensors in CANoe

In **CtApMySWC.c**

- React on incoming information about front doors (opened or closed)
- Turn the front interior light on or off based on doors state

in **CtSaInteriorLight.c**

- Check received data by setting a breakpoint during debugging

- Build the ECU code in Visual Studio by pressing F7

Exercise 1 – Software Components
Part 4/4: Generate code, program Runnables and test it

Start CANoe with double click on the _E1_SoftwareComponents\CANoe\MyECU.cfg

When CANoe is opened, switch back to the Microsoft Visual Studio.
Do not start the CANoe measurement yet.

Press **F5** in Visual Studio to start Debug session and to attaches to CANoe
(via RuntimeKernel.exe process)

Switch back to CANoe and start the simulation by pressing **F9**

Exercise 1 – Software Components
Part 4/4: Generate code, program Runnables and test it

Set a breakpoint in the runnable **RCtApMySwcCode** and test whether the reception of the information about door state triggers the runnable and whether the values of the door states are correct.

Set a breakpoint in the runnable **RCtSaInteriorLightSwitchLight** and test whether the reception of the information "light on" triggers the runnable and whether the value of the light is correct.

Take a look how and from where Runables are executed. Find definitions of **Rte_Read** and **Rte_Write** functions and try to understand how they work…

What is difference between **Rte_Read** and **Rte_IRead**?