Learn-/Training Document

Siemens Automation Cooperates with Education (SCE) | From NX MCD V12/TIA Portal V15.0

DigitalTwin@Education Module 150-006
Signal Creation for a Dynamic 3D Model in the Mechatronics Concept Designer CAE System

siemens.com/sce

SIEMENS

Global Industry Partner of WorldSkills International

worldskills

Matching SCE trainer packages for this Learn-/Training Document

SIMATIC STEP 7 Software for Training (incl. PLCSIM Advanced)

- **SIMATIC STEP 7 Professional V15.0 - Single License**
  Order no.: 6ES7822-1AA05-4YA5
- **SIMATIC STEP 7 Professional V15.0 - Classroom License for 6 Users**
  Order no.: 6ES7822-1BA05-4YA5
- **SIMATIC STEP 7 Professional V15.0 - Upgrade License for 6 Users**
  Order no.: 6ES7822-1AA05-4YE5
- **SIMATIC STEP 7 Professional V15.0 - Student License for 20 Users**
  Order no.: 6ES7822-1AC05-4YA5

SIMATIC WinCC Engineering/Runtime Advanced software in the TIA Portal

- **SIMATIC WinCC Advanced V15.0 - Classroom License for 6 Users**
  6AV2102-0AA05-0AS5
- **Upgrade SIMATIC WinCC Advanced V15.0 - Classroom License for 6 Users**
  6AV2102-4AA05-0AS5
- **SIMATIC WinCC Advanced V15.0 - Student License for 20 Users**
  6AV2102-0AA05-0AS7

NX V12.0 Educational Bundle (Schools, universities, not for in-company training centers)

- **Contact**: academics.plm@siemens.com

Additional information regarding SCE

siemens.com/sce

Information regarding use

We wish to thank the HS Darmstadt, particularly Mr. Heiko Webert and Mr. Prof. Dr.-Ing. Stephan Simons, and all other persons involved for their support in the preparation of this SCE learning/training document.

Table of contents

List of figures

List of tables

Signal Creation for a Dynamic 3D Model in the Mechatronics Concept Designer CAE System

1 Goal

In Module 4 of the DigitalTwin@Education workshop series, you created the static 3D model of a sorting plant entirely on your own. The result was an assembly in which the required individual components of the sorting plant are inserted and positioned correctly in space. Building on that, Module 5 was concerned with dynamization of the 3D model. By assigning physical properties, the components of the sorting plant are able to interact with one another.

In order for the digital twin to work in interaction with a virtual PLC, what is finally required is a connection between the Mechatronics Concept Designer (MCD) and PLCSIM Advanced – the virtual PLC that will be used to simulate the PLC with the automation program. The goal of this module is to create signals and to map them to the two programs. You will then use the automation program from Module 1 of the DigitalTwin@Education series to validate the correct functioning of your digital twin.

2 Requirement

Knowledge of the dynamic properties of the model used in Module 5 is required for this module. In addition, you should understand how the automation program works (a result of having completed Modules 1 – 2), as you will be using the program again in this module.

3 Required hardware and software

The following components are required for this module:

1 **Engineering station**: Requirements include hardware and operating system (for additional information: see Readme on the TIA Portal Installation DVDs and in the NX software package)

2 **SIMATIC STEP 7 Professional software in TIA Portal** – V15.0 or higher

3 **SIMATIC WinCC Runtime Advanced software in TIA Portal** – V15.0 or higher

4 **SIMATIC S7-PLCSIM Advanced software** – V2.0 or higher

5 **NX software with Mechatronics Concept Designer application** – V12.0 or higher

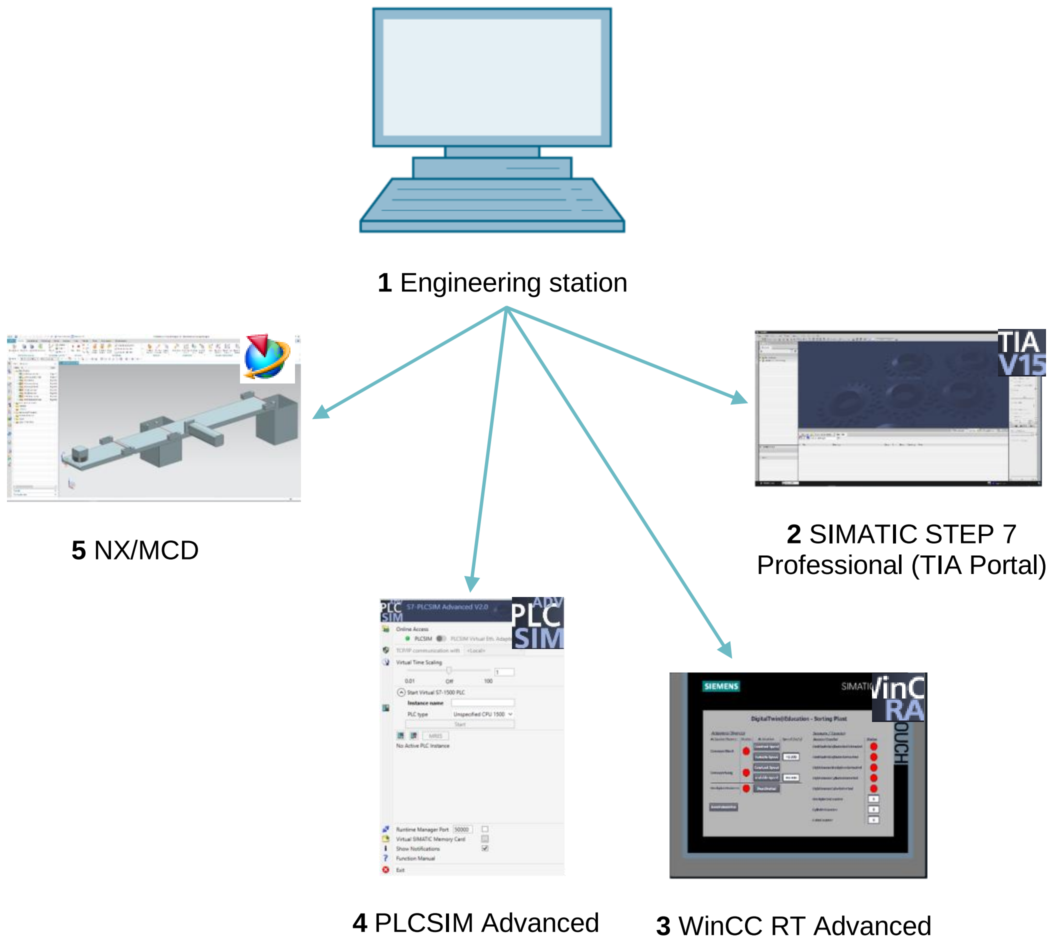


Figure 1: Overview of required software and hardware components in this module

Figure 1 clearly shows that the engineering station is the only hardware component of the system. The remaining components are based exclusively on software.

4 Theory

4.1 Communication with external sources

In Module 5 of this workshop series, you created dynamic properties and tested them for proper functioning using the Runtime Inspector in Mechatronics Concept Designer. For a digital twin, however, it is useful to establish a connection to a PLC so that the PLC can change dynamic properties in MCD and results from MCD can be made available to the PLC.

MCD provides a variety of ways to communicate with external programs (see Chapter 9, Link [1]).

These include:

- Communication with MATLAB using the MATLAB protocol
- Connection to an OPC server (as of MCD V12.01, also to an OPC UA server)
- S7 communication via PLCSIM Advanced or directly using the PROFINET protocol
- Connection to *shared memory*, for example, for SIMIT
- TCP/UDP connections

In this module, you will configure communication with PLCSIM Advanced, as was already used in Modules 1 – 3 of this workshop series. This allows a virtual PLC to be connected to NX/MCD.

4.2 Signal properties in Mechatronics Concept Designer

Communication with external programs is configured in the NX Mechatronics Concept Designer application through the definition and mapping signals.

The workspace of Mechatronics Concept Designer is shown in Figure 2. To open this application in NX, use the familiar Command Finder at the top right of the screen to search for the "**Mechatronics Concept Designer**" application.

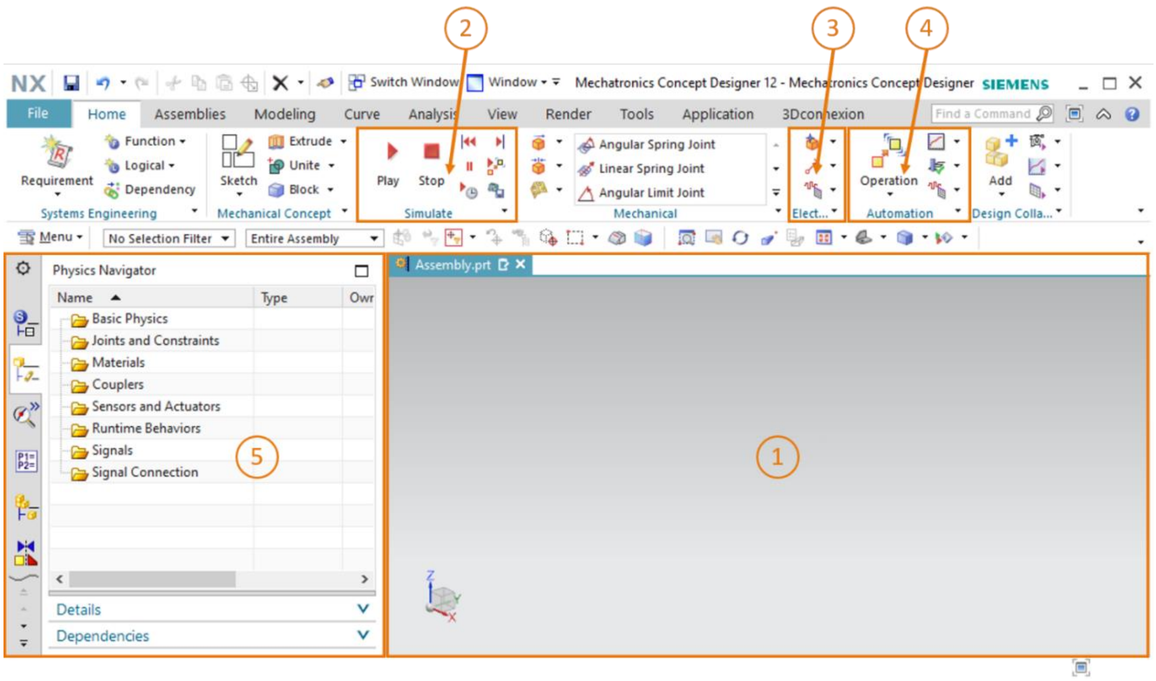


Figure 2: "Mechatronics Concept Designer" application in NX with markings for explanations of areas in the text

The following windows are used in this application for the definition of signals and for testing the digital twin:

- The central screen (see Figure 2, area 1) contains the three-dimensional graphics window, where, you can track the functioning of your dynamic 3D model in interaction with the virtual PLC during a simulation.
- You can control the simulation of your model from a menu group located in the middle of the menu bar (see Figure 2, area 2). You will make use of these functions in Chapter 7.3.

- You will find signal properties from the field of electrics in the menu group next to the dynamic mechanical properties (see Figure 2, area 3). Here, you can create signals and tables. Several of the commands are explained briefly below.
  - You can use the **Signal** command to create a signal in your model for controlling physical properties of an object with a runtime expression. A runtime expression is a non-static value that can change at runtime of a simulation. This expression is connected internally to MCD or it is determined by a connection to a signal from an external source, such as PLCSIM Advanced.
  - By creating a **Symbol Table**, you define a list of symbols that are used for unambiguous naming of signals. You can also import a symbol table from an external source, such as STEP 7.
  - Signals and runtime expressions can be interconnected using the **Signal Adapter**. It is possible to use multiple signals and runtime expressions for each signal adapter. In addition, both signals and runtime expressions can be created using this command.
- The signal properties from the field of automation can also be found in the menu bar of MCD (see Figure 2, area 4). Here, you will make use of the following property:
  - **Signal Mapping** interconnects signals from MCD with signals from external programs. An example of an external program is PLCSIM Advanced.
- From the Resource bar on the left part of the screen in MCD (see Figure 2, area 5), you can open the Physics Navigator tab, among others. This is where, your signals and connections will be stored.

NOTE

For further information on the signal properties in Mechatronics Concept Designer, you can search for corresponding entries in the online help (see Chapter 9, Link [2]).

It is recommended that the search be conducted using English terms since German entries are incomplete.

5 Task

In the following, you are to add signals to the dynamic 3D model of the sorting plant you created in Module 5 and establish a connection to a virtual PLC. In addition, you are to download the automation program from Module 1 of the DigitalTwin@Education workshop series to the virtual PLC and independently validate you own digital twin.

You will once again need the Mechatronics Concept Designer (MCD) NX application for this. This time, however, your sole focus is on connecting your dynamic 3D model to external programs.

6 Planning

The **NX** CAD system in version **V12.0 or higher** is required for the mapping of signals for a dynamic 3D model and its commissioning. The **Mechatronics Concept Designer (MCD)** add-on module must also be available in NX.

You must have knowledge of **static** and **dynamic 3D models**, which you can acquire by completing Modules 4 and 5. You should also brush up on your knowledge on **how the automation program works** from **Modules 1 – 2** of the DigitalTwin@Education series. If you are unclear about how the sorting plant works, you should especially review the theory section in **Chapter 4.2** of **Module 1** again.

Also reacquaint yourself with the **interaction between the virtual PLC and the digital twin** from **Module 1** and keep the description of Module 1 of this workshop series at hand as you will need it, especially in Chapters 7.2 and 7.3.

The Siemens "**Guide to Standardization**" has been followed for naming the signals. You can find this guide by going to Chapter 9 and selecting the link there [3].

7 Structured step-by-step instructions

The folder "**150-006_DigitalTwinAtEducation_NX_dynModelSignals**" is made available with this module. The folder has three subfolders:

- "**fullDynModel**" contains the full dynamic 3D model of the sorting plant from **Module 5.** You can use this model as a starting point for this module if your results from Module 5 are incomplete.
- "**fullDigTwin**" contains the full digital twin and thus the solution to this module. It serves as an aid in case you need help completing a step.
- "**fullPlcBasic**" provides the automation program with integrated HMI with which you are familiar from **Module 1**. You will need this to test your digital twin.

As a reminder, use the Command Finder if you are unable to find a command or an application in the development environment at any time in this module. This is located in the upper right part of the NX user interface screen, as shown in Figure 3.

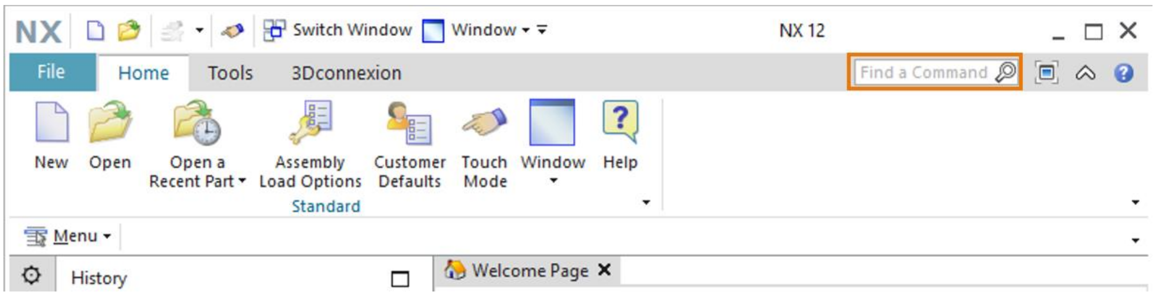


Figure 3: Command Finder in the NX menu, highlighted in orange

You can select the appropriate command from the search results. NX also shows you where the command is located so that you can also select it directly from the menu in the future.

IMPORTANT: The user interface and the arrangement of various commands in the menus change with new versions of NX. In addition, users can define their own user interfaces. While the following descriptions depict the standard user interface of NX12.0, the interface in your version may be different. **For this reason, if you are unable to find a command in the window at the positions described, use the Command Finder.**

Also bear in mind that this description only represents a suggested solution. The aim here was to describe a straightforward procedure by which your digital twin can be easily made to interact with a virtual PLC from Modules 1 - 3.

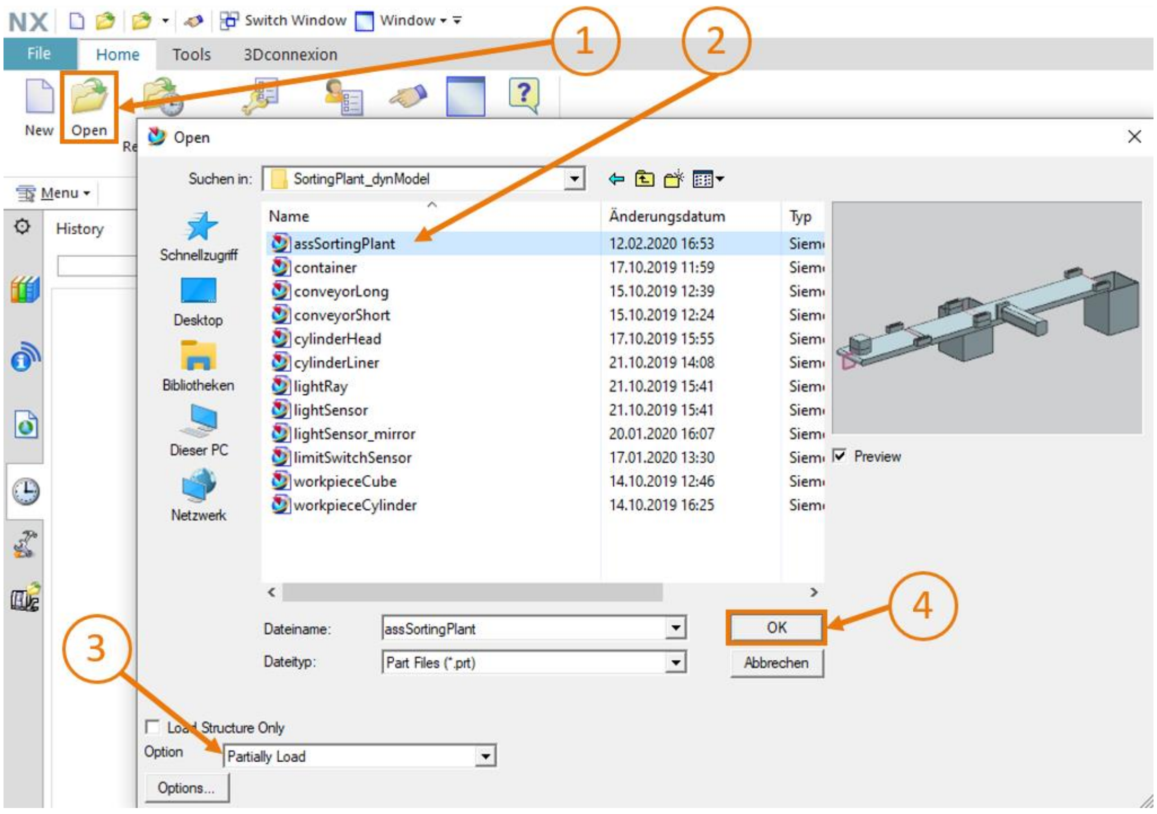
Note that certain passages in this module are labeled as "Sections". Because frequent reference is made to these passages in the course of the description, this labeling is intended as an orientation guide.

7.1 Creating the signals for the dynamic model

In this chapter, you are to create all the required signals for your sorting plant, which are to be externally controllable by a PLC. Proceed as follows for this:

→ Use your operating system to copy your models that you used in Module 5. Save them in a new folder of your file system. As mentioned in Chapter 7, if you have an incomplete dynamic model, you can also access the "**fullDynModel**" project and create a working copy from this folder.

→ Start NX and wait until the program opens and you see the Home page. Click the "**Open**" button (see Figure 4, step 1) and navigate to the folder you created beforehand. You now see the parts used in Module 5. Select the "**assSortingPlant**" assembly, which contains the full dynamic 3D model of the sorting plant (see Figure 4, step 2). Select the "**Partially Load**" option (see Figure 4, step 3) so that only the models and dynamic properties of the individual components of the assembly are loaded but not any additional drawings or coordinate systems. Finally, confirm your selection by clicking on "**OK**" (see Figure 4, step 4).



Suchen in: SortingPlant_dynModel

Name	Änderungsdatum	Typ
assSortingPlant	12.02.2020 16:53	Siem
container	17.10.2019 11:59	Siem
conveyorLong	15.10.2019 12:39	Siem
conveyorShort	15.10.2019 12:24	Siem
cylinderHead	17.10.2019 15:55	Siem
cylinderLiner	21.10.2019 14:08	Siem
lightRay	21.10.2019 15:41	Siem
lightSensor	21.10.2019 15:41	Siem
lightSensor_mirror	20.01.2020 16:07	Siem
limitSwitchSensor	17.01.2020 13:30	Siem
workpieceCube	14.10.2019 12:46	Siem
workpieceCylinder	14.10.2019 16:25	Siem

Dateiname: assSortingPlant
Dateityp: Part Files (*.prt)
Load Structure Only
Option: Partially Load

Figure 4: Opening an assembly in NX

Section: Creation and connection of signals with the signal adapter

→ Once the assembly has opened in the NX **"Mechatronics Concept Designer"** application, you create your first example. For this, you are to create and connect a signal for activating the object source, which is required for generating cuboid workpieces. To add signals and interconnect their dynamic properties, first open the "**Signal Adapter**" command from the **"Electrical"** menu group, as shown in Figure 5, step 1. The **"Signal Adapter"** command window then appears. Start by selecting a parameter of a dynamic property that is to be connected to a signal. For this, click on the "**Select Physics Object**" button in the "**Parameters**" group (see Figure 5, step 2). Navigate in the Resource bar to the "**Physics Navigator"** tab (see Figure 5, step 3) and there, select the object source "**osWorkpiece Cube**" as the first parameter (see Figure 5, step 4). Once you have selected this, you can now select the corresponding parameter you want to assign to a signal from the "**Parameter Name**" list in the command window. In this case, use the parameter name "**active**" of the selected object source (see Figure 5, step 5). Click on the "**Add Parameter**" button (see Figure 5, step 6), which adds the parameter to this signal adapter.

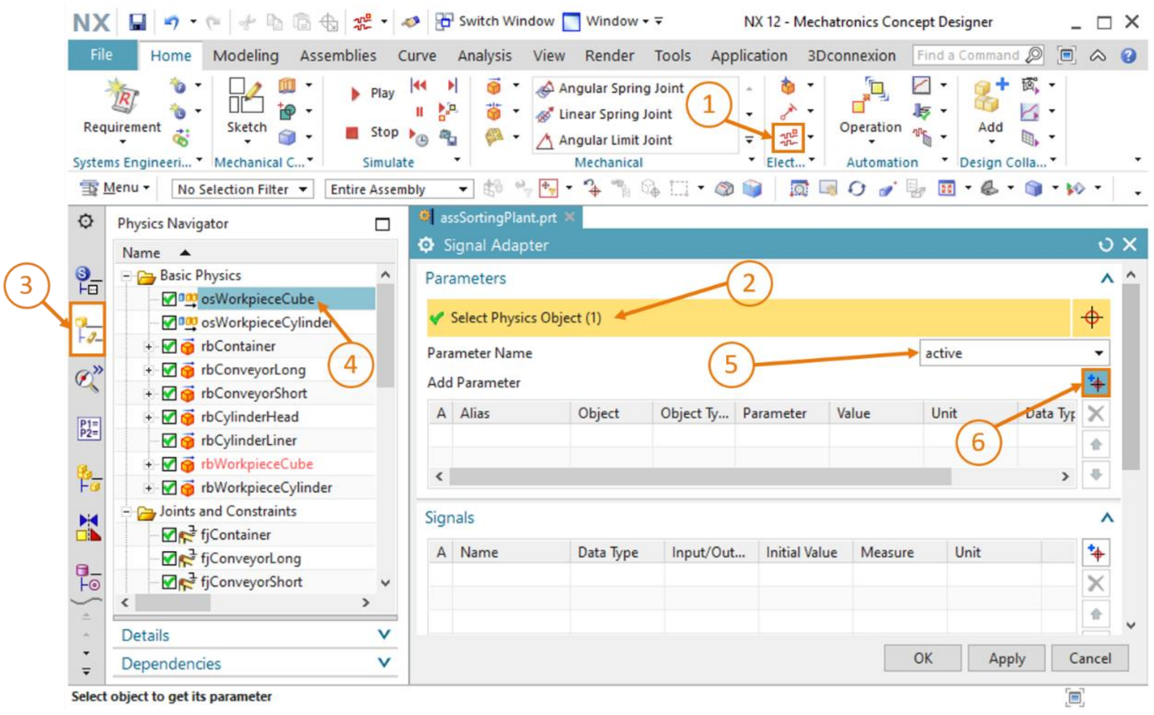
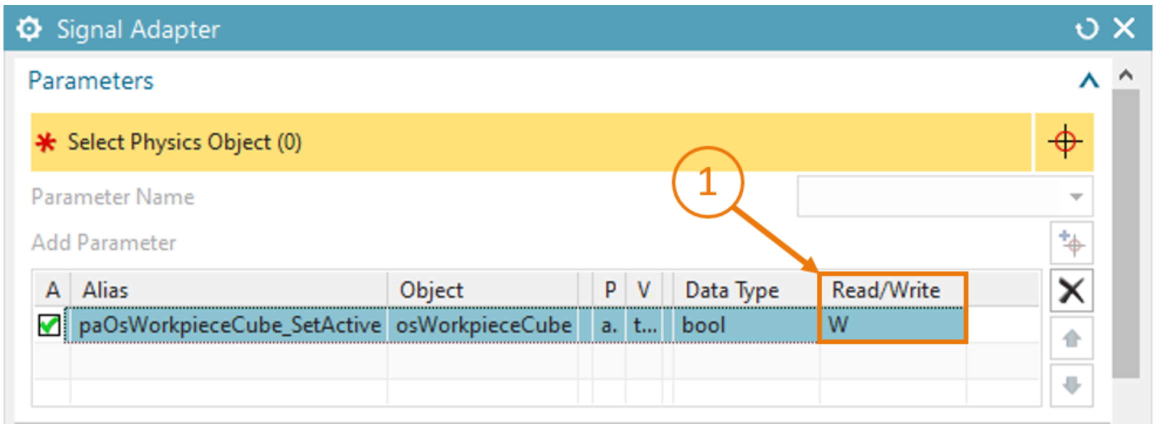


Figure 5: Adding parameters of dynamic properties for signals in the signal adapter

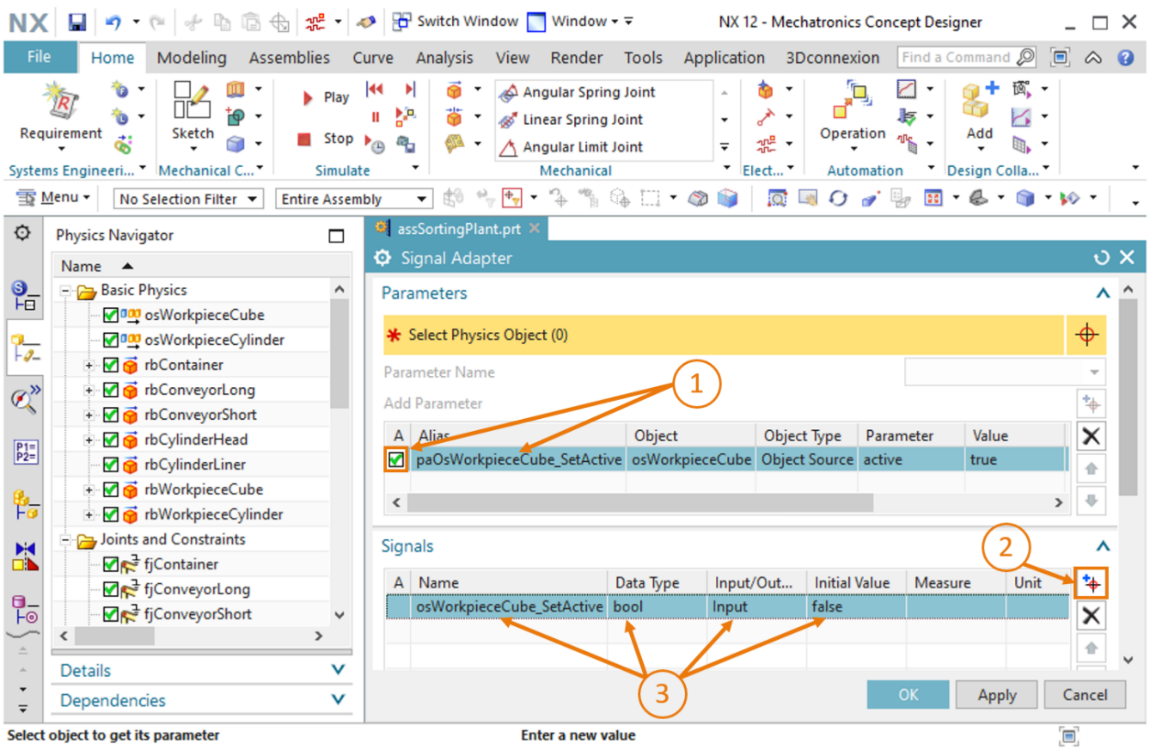
→ The parameter you have just selected can then be seen in the table in the "**Parameters**" group. Change its alias name to "**paOsWorkpieceCube_SetActive**" (see Figure 7, step 1). The prefix **"pa"** is meant to stand for **"parameter"** so as to have a clear distinction from a signal name. Also click on the box at the beginning of the table so that you can later assign a signal to the parameter. This is indicated by a check mark ✓. If you scroll to the right in the "**Parameters**" group, you can see other properties of the parameter, such as the **"Read/Write"** property, which indicates whether a parameter can be read (**"R"** for Read) or written **("W"** for Write). The current parameter **"paOsWorkpieceCube_SetActive"** can only be written (see Figure 6, step 1).



A	Alias	Object	P	V	Data Type	Read/Write
✓	paOsWorkpieceCube_SetActive	osWorkpieceCube	a.	t...	bool	W

Figure 6: Read/Write property of a parameter

→ Now you still need an associated signal to which the parameter is to be connected. For this, click on the "**Add**" button in the "**Signals**" group (see Figure 7, step 2). A new signal appears. Adapt the properties of the signal in line with the parameter. To do this, double-click on the default name of the signal **"Signal_0"** and rename it "**osWorkpieceCube_SetActive**". The same data type must be selected for the signal and for the associated parameter. That is "**bool**" in this case. The value of the **"Input/Output"** property must be selected in accordance with the **"Read/Write"** property of the parameter. For a parameter that must be written, the signal must be an input from an external source from the perspective of MCD. For a parameter that will be read, the signal must be an output to an external program. Because the current parameter **"paOsWorkpieceCube_SetActive"** will be written, the "**Input**" value must be selected for the **"osWorkpieceCube_SetActive"** signal. The selected initial value in the table must be equated with the initial value of the signal. In this case, this is "**false**" (see Figure 7, step 3).



A	Alias	Object	Object Type	Parameter	Value
✓	paOsWorkpieceCube_SetActive	osWorkpieceCube	Object Source	active	true

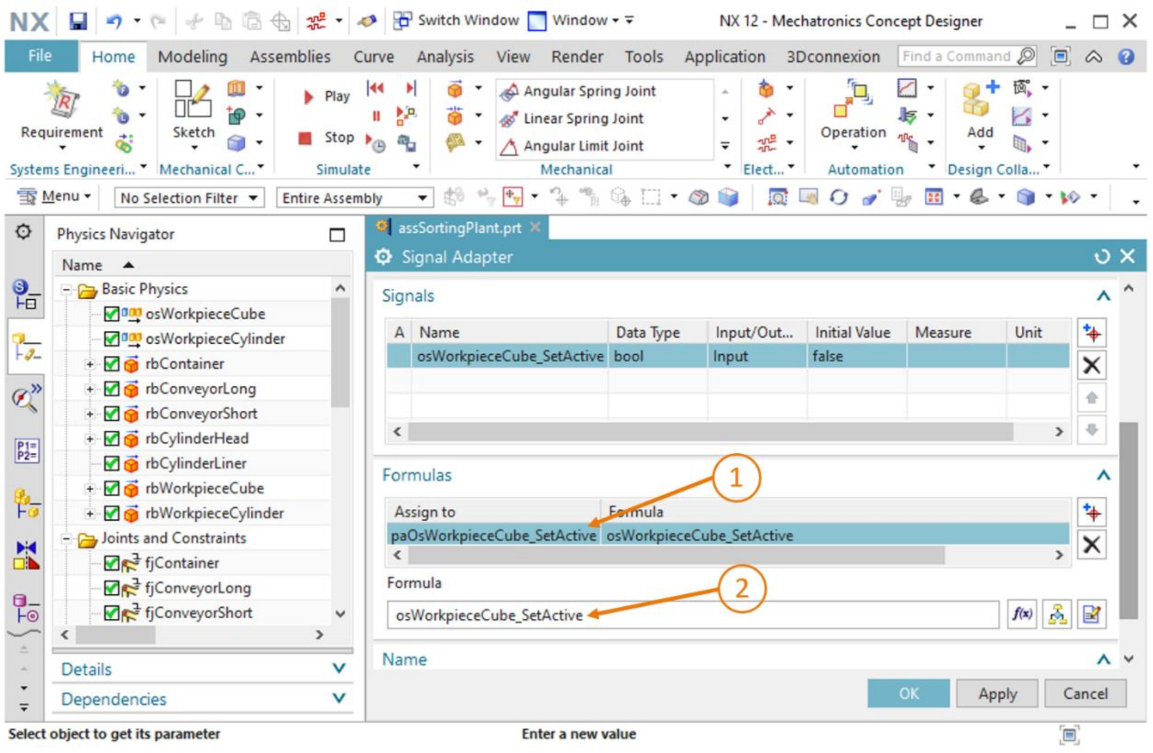
A	Name	Data Type	Input/Out...	Initial Value	Measure	Unit
	osWorkpieceCube_SetActive	bool	Input	false		

Select object to get its parameter

Enter a new value

Figure 7: Creating a suitable signal for a parameter

→ The parameter and its associated signal must now be logically interconnected. To do this, scroll down in the command window to the **"Formulas"** command group. There, you see that you can assign a formula to the **"paOsWorkpieceCube_SetActive"** parameter. For this, click on the corresponding row in the table (see Figure 8, step 1). You can now make a suitable assignment in the **"Formula"** box. In this case, a simple assignment of the "**osWorkpieceCube_SetActive**" signal to the parameter is sufficient, as shown in Figure 8, step 2. After pressing the Enter key on your keyboard, you see the assignment you just made in the "**Formula**" column of the table.



Signals

A	Name	Data Type	Input/Out...	Initial Value	Measure	Unit
	osWorkpieceCube_SetActive	bool	Input	false		

Formulas

Assign to	Formula
paOsWorkpieceCube_SetActive	osWorkpieceCube_SetActive

Formula: osWorkpieceCube_SetActive

Figure 8: Definition of a formula between signal and parameter

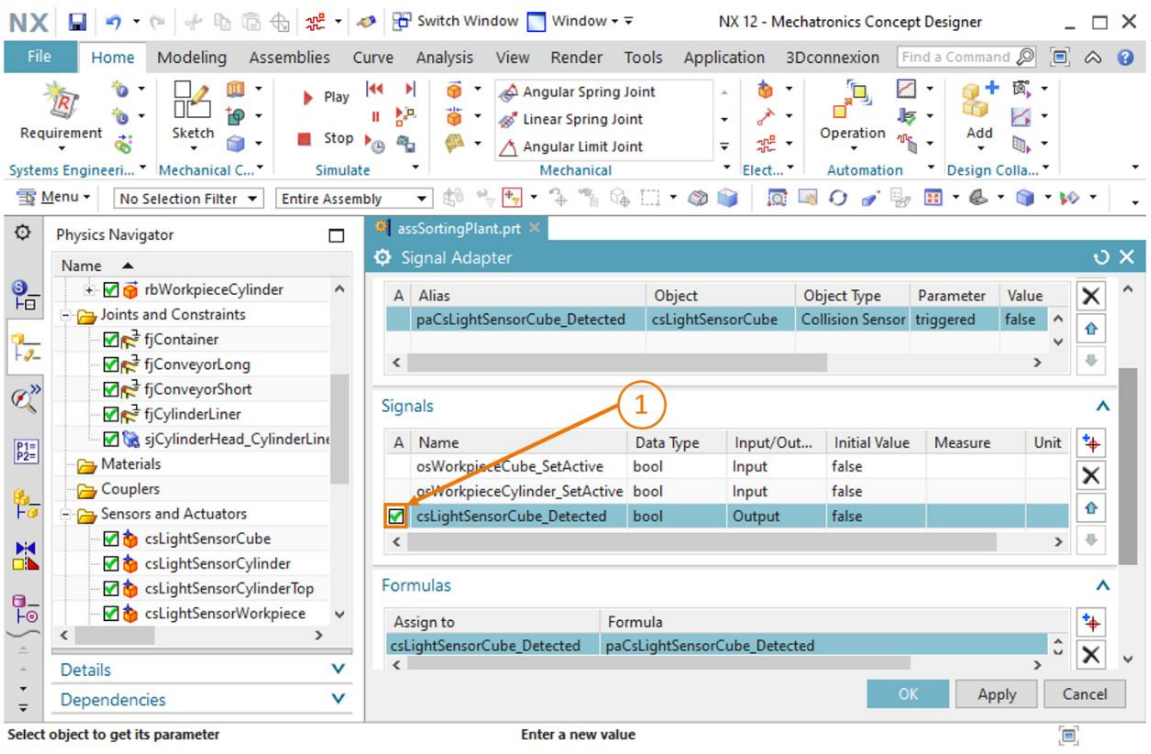
NOTE

Complex formulas that are dependent on multiple parameters and/or signals can also be selected. But in the interest of traceability of the digital twin, care should be taken to define formulas in the signal adapter that are as simple as possible. Priority should always be given to making the logic part of the automation program and not part of the digital twin.

You have now created the first signal of your digital twin on your own. Now create the remaining signals following the procedure in Chapter 7.1, "**Section: Creation and connection of signals with the signal adapter**". Use the following specifications for this:

→ The "**active**" parameter with alias name "**paOsWorkpieceCylinder_SetActive**" is to be created in the signal adapter from the "**osWorkpieceCylinder**" object source. Again, select the check box ✓ in front of the parameter so that a signal can be assigned. The associated signal is to be named "**osWorkpieceCylinder_SetActive**" and have data type "**bool**". This signal must be defined as "**Input**" and have an output value of "**false**". For the formula for the **"paOsWorkpieceCylinder_SetActive"** parameter, use the direct assignment of the "**osWorkpieceCylinder_SetActive**" signal.

→ For the "**triggered**" parameter of the "**csLightSensorCube**" collision sensor, create a new parameter with alias name "**paCsLightSensorCube_Detected**" in the signal adapter. For the associated signal, specify a **Boolean** signal named "**csLightSensorCube_Detected**". This must be defined as "**Output**" because the **"paCsLightSensorCube_Detected"** parameter has the value "**R**" in the **"Read/ Write"** column, which means it must be read. Assign an output value of "**false**" to the signal. Select the check box ✓ in front of the signal, as shown in Figure 9, step 1, so that a formula can be assigned to the output signal. Use the direct assignment of the "**paCsLightSensorCube_Detected**" parameter as the formula for the **"csLightSensorCube_Detected"** signal.



A	Alias	Object	Object Type	Parameter	Value
	paCsLightSensorCube_Detected	csLightSensorCube	Collision Sensor	triggered	false

Signals

A	Name	Data Type	Input/Out...	Initial Value	Measure	Unit
	osWorkpieceCube_SetActive	bool	Input	false		
	osWorkpieceCylinder_SetActive	bool	Input	false		
✓	csLightSensorCube_Detected	bool	Output	false		

Formulas

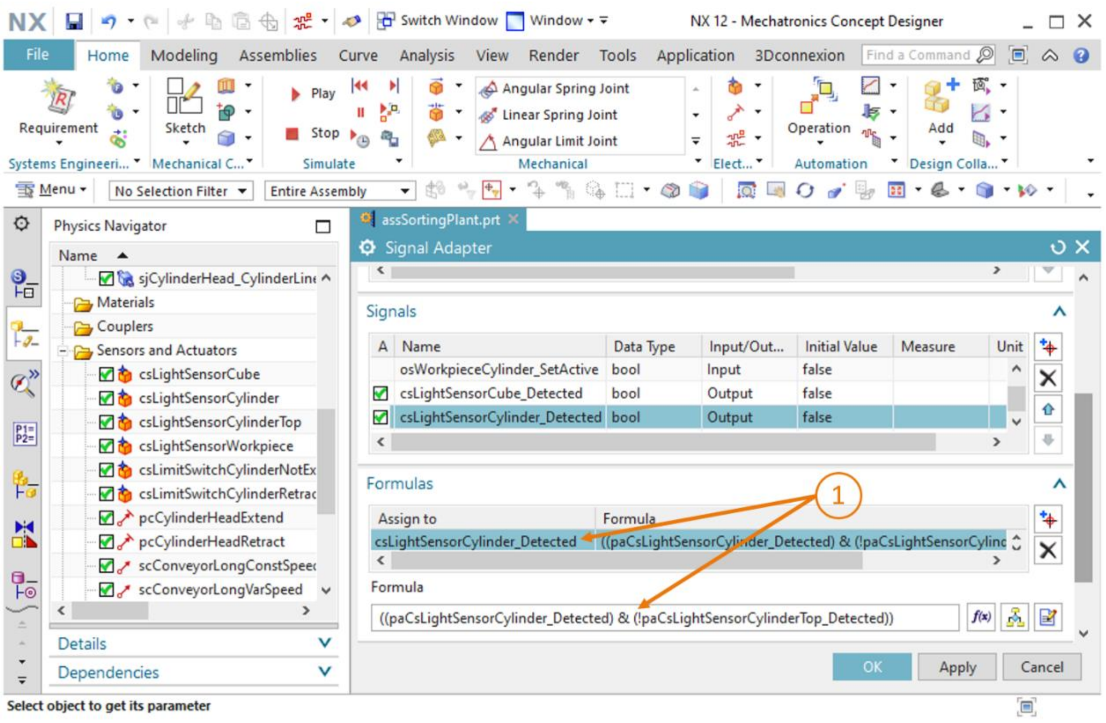
Assign to	Formula
csLightSensorCube_Detected	paCsLightSensorCube_Detected

Figure 9: Creating an output signal for a light sensor

→ The next light sensor system in the middle of the conveyor process consists of two collision sensors. Two parameters must therefore be created for this. First, for the "**triggered**" parameter of the "**csLightSensorCylinder**" collision sensor, define a parameter with alias name "**paCsLightSensorCylinder_Detected**" in the signal adapter. For the "**triggered**" parameter of the second "**csLightSensorCylinderTop**" collision sensor, define a parameter with alias name "**paCsLightSensorCylinderTop_Detected**" in the signal adapter. A combined signal that reacts to both parameters is now to be created. This signal should be named "**csLightSensorCylinder_Detected**" and have data type "**bool**". Because both of the above-named parameters have to be read in this case as well, the combined signal is to be configured as "**Output**" and have the initial value "**false**". Specify the following formula for **"csLightSensorCylinder_Detected"**, as shown in Figure 10, step 1:

"**((paCsLightSensorCylinderDetected) & (!paCsLightSensorCylinderTop_Detected))**".

This formula constitutes an AND operation of the two parameters in which the second parameter is negated, i.e. output signal **"csLightSensorCylinder_Detected"** becomes **"true"** when **"paCsLightSensorCylinderDetected"** assumes the **"true"** value and **"paCsLightSensorCylinderTop_Detected"** is simultaneously **"false"**. According to the explanation of the operation of the sorting plant in Module 1 of this workshop series, the different height of the workpieces results in a cylindrical workpiece only being detected when the lower light sensor of this light sensor system trips and, at the same time, the upper light sensor detects no collision. This logic is represented with this formula.



Signals

A	Name	Data Type	Input/Out...	Initial Value	Measure	Unit
	osWorkpieceCylinder_SetActive	bool	Input	false		
✓	csLightSensorCube_Detected	bool	Output	false		
✓	csLightSensorCylinder_Detected	bool	Output	false		

Formulas

Assign to	Formula
csLightSensorCylinder_Detected	((paCsLightSensorCylinder_Detected) & (!paCsLightSensorCylinc

Formula
((paCsLightSensorCylinder_Detected) & (!paCsLightSensorCylinderTop_Detected))

Select object to get its parameter

Figure 10: Formula for the signal of the "csLightSensorCylinder" light sensor system

NOTE

When an input signal is to be assigned to a writable parameter, the check box in front of the parameter must be selected ✓. Likewise, as soon a readable parameter is to be assigned to an output signal, the check box in front of the signal must be selected ✓.

You can only define the corresponding formula when a check mark ✓ is visible.

→ For the "**triggered**" parameter of the "**csLightSensorWorkpiece**" collision sensor, you need a parameter to which you should give the alias name "**paCsLightSensor Workpiece_Detected**". To do this, create a **Boolean** signal named "**csLightSensor Workpiece_Detected**". Also, define this as an **output** signal with an initial value of "**false**". The formula for the **"csLightSensorWorkpiece_Detected"** signal is "**paCsLightSensor Workpiece_Detected**".

→ For the "**triggered**" parameter of the "**csLimitSwitchCylinderNotExtended**" collision sensor, define a parameter with alias name "**paCsLimitSwitchCylinderNotExtended_ Activated**" in the signal adapter. Add a **Boolean** signal named "**csLimitSwitchCylinderNot Extended_Activated**". This signal is to be defined as an **output**-type signal and have an initial value of "**false**". As the formula for the **"csLimitSwitchCylinderNotExtended_ Activated"** signal, specify the assignment "**paCsLimitSwitchCylinderNotExtended_ Activated**".

→ The "**triggered**" parameter of the last remaining collision sensor "**csLimitSwitch CylinderRetracted**" is also to be created as a parameter in the signal adapter with the alias name "**paCsLimitSwitchCylinderRetracted_Activated**". The associated "**csLimitSwitch CylinderRetracted_Activated**" signal is to have the "**bool**" data type. In addition, it is to be defined as an **output** signal with an initial value of "**false**". The formula for **"csLimitSwitchCylinderRetracted_Activated"** is to be specified as a simple assignment of the "**paCsLimitSwitchCylinderRetracted_Activated**" parameter.

→ For the "**active**" parameter of the "**pcCylinderHeadExtend**" position control, a parameter with alias name "**paPcCylinderHeadExtend_SetActive**" is to be created in the signal adapter. Then create a new signal named "**pcCylinderHeadExtend_SetActive**" of data type "**bool**". Define the signal as "**Input**" with an initial value of "**false**". As a formula for the **"paPcCylinderHeadExtend_SetActive"** parameter, specify the assignment of the "**pcCylinderHeadExtend_SetActive**" signal.

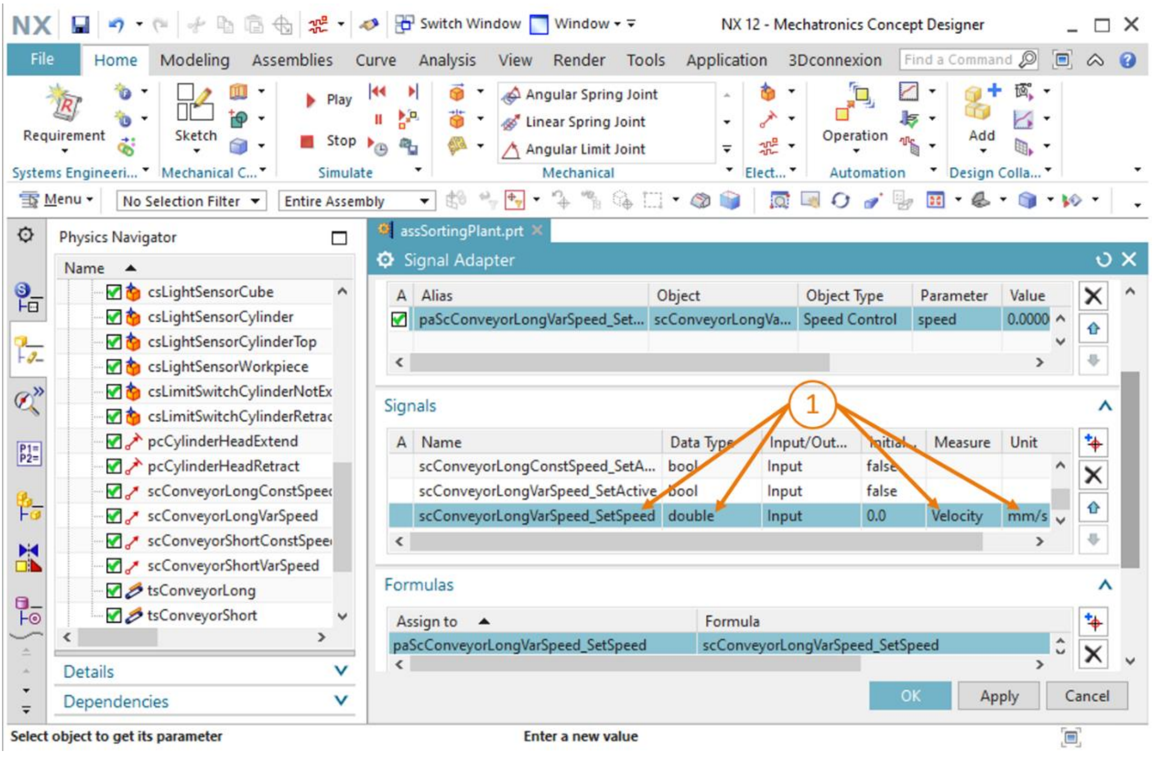
→ For the "**active**" parameter of the "**pcCylinderHeadRetract**" position control, you are to create a parameter with alias name "**paPcCylinderHeadRetract_SetActive**" in the signal adapter. The signal corresponding to this is to be named "**pcCylinderHead Retract_SetActive**". Define this signal as **Boolean** and as an "**Input**" signal with an initial value of "**false**". The formula of **"paPcCylinderHead Retract_SetActive"** is "**pcCylinder HeadRetract_SetActive**".

→ The "**active**" parameter of the "**scConveyorLongConstSpeed**" speed control requires a parameter in the signal adapter with alias name "**paScConveyorLong ConstSpeed_SetActive**". The associated signal is to be named "**scConveyorLong ConstSpeed_SetActive**". This signal is to have "**bool**" data type, be defined as an **output** signal and have an initial value of "**false**". A simple assignment of "**scConveyorLongConst Speed_SetActive**" is to be used as the formula for **"paScConveyorLong ConstSpeed_SetActive"**.

→ For the "**scConveyorLongVarSpeed**" speed control, two parameters and two signals must be defined in the signal adapter.

Of these, the first parameter and its associated signal in the signal adapter are to be used to activate the speed control. For this, create a new parameter with alias name "**paScConveyor LongVarSpeed_SetActive**" for the "**active**" parameter of the "**scConveyorLongVarSpeed**" speed control. Also, create the "**scConveyorLongVar Speed_SetActive**" signal and declare the data type as "**bool**". The signal is to serve as an **input**. The output value is to be "**false**". Finally, specify the "**scConveyorLongVar Speed_SetActive**" signal as the formula for "**paScConveyorLongVarSpeed_SetActive"**.

The purpose of the second signal is to allow the setting of a variable setpoint speed of the speed control. For this, create a new parameter in the signal adapter that is to be connected with the "**speed**" parameter of the "**scConveyorLongVarSpeed**" speed control, and use the alias name "**paScConveyorLongVarSpeed_SetSpeed**". The associated signal is to be named "**scConveyorLongVarSpeed_SetSpeed**". Because a speed is to be specified with this signal, the data type must be declared as "**double**". For non-"**bool**"-type signals, the physical type must be entered under **"Measure"** and the associated physical unit of the value under **"Unit"**. In this case, be sure to specify the following for the current signal: the value "**speed**" in the **"Measure"** column and the expression "**mm/s**" in the **"Unit"** column (see Figure 11, step 1). This signal is also an **input** signal. Also specify "**0.0**" as the initial value. The assignment of the "**scConveyorLongVarSpeed_SetSpeed**" signal is sufficient as the formula for the "**paScConveyorLongVarSpeed_SetSpeed**" parameter.



A	Alias	Object	Object Type	Parameter	Value
✓	paScConveyorLongVarSpeed_Set...	scConveyorLongVa...	Speed Control	speed	0.0000

A	Name	Data Type	Input/Out...	Initial...	Measure	Unit
	scConveyorLongConstSpeed_SetA...	bool	Input	false		
	scConveyorLongVarSpeed_SetActive	bool	Input	false		
	scConveyorLongVarSpeed_SetSpeed	double	Input	0.0	Velocity	mm/s

Assign to	Formula
paScConveyorLongVarSpeed_SetSpeed	scConveyorLongVarSpeed_SetSpeed

Figure 11: Creating a speed signal of the "double" data type

→ For the "**scConveyorShortConstSpeed**" speed control, the "**active**" parameter is to be inserted as a new parameter in the signal adapter. Give this parameter the alias name "**paScConveyorShortConstSpeed_SetActive**". Create the corresponding "**scConveyor ShortConstSpeed_SetActive**" signal of data type "**bool**". It is to be designated as an **input** signal have an initial value of "**false**". Finally, as the formula for **"paScConveyorShort ConstSpeed_SetActive"**, use a direct assignment of the "**scConveyorShort ConstSpeed_SetActive**" signal.

→ The "**scConveyorShortVarSpeed**" speed control also requires two signals in the signal adapter.

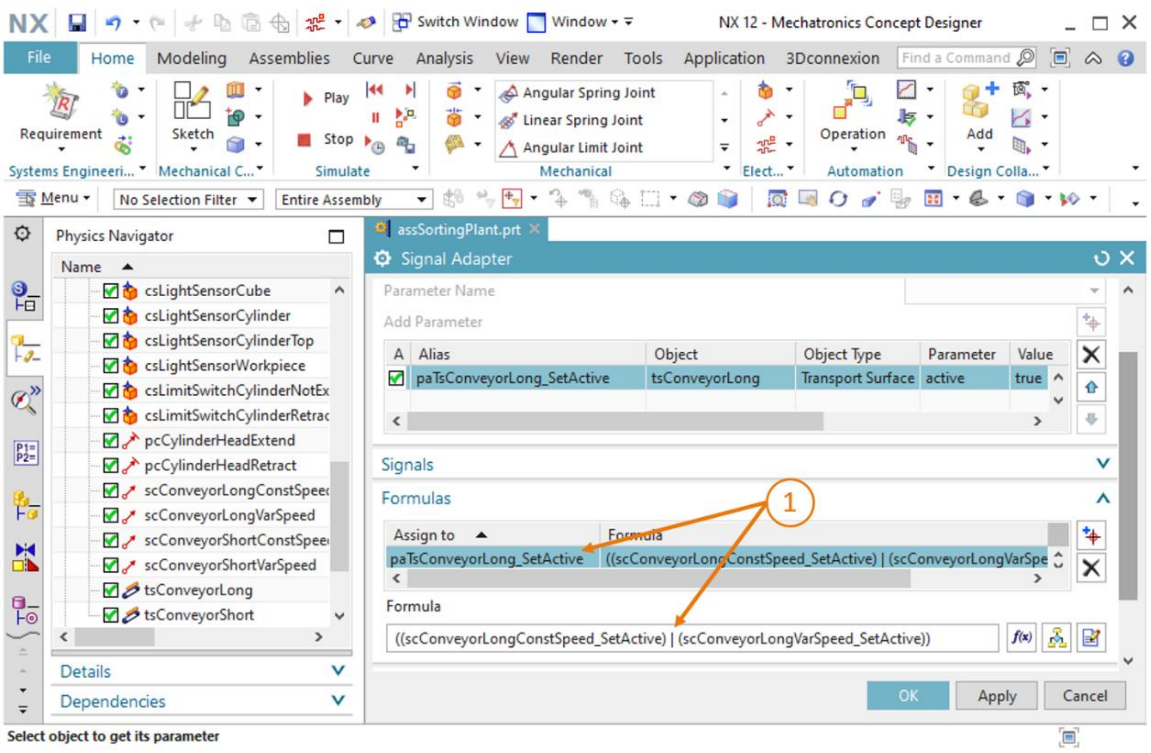
The first parameter in the signal adapter is to relate to the "**active**" parameter of the **"scConveyorShortVarSpeed"** speed control and have the alias name "**paScConveyor ShortVarSpeed_SetActive**". A **Boolean** signal named "**scConveyorShort VarSpeed_ SetActive**" is to be created as the associated signal. This is to be an **input** signal and begin with an initial value of "**false**". The "**scConveyorShortVarSpeed_SetActive**" signal is to be assigned to the **"paScConveyorShortVarSpeed_SetActive"** parameter.

The second signal is to be used to set the setpoint speed for the speed control. For this, create a new parameter in the signal adapter that is based on the "**speed**" parameter of the **"scConveyorShortVarSpeed"** speed control. Give this parameter the alias name "**paScConveyorShortVarSpeed_SetSpeed**". When defining the new "**scConveyorShort VarSpeed_SetSpeed**" signal, be sure to specify "**double**" as the data type, "**speed**" in the "**Measure**" column and "**mm/s**" in the "**Unit**" column. This signal is an **input** signal with an initial value of "**0.0**". The formula for the **"paScConveyorShortVarSpeed_SetSpeed"** parameter is "**scConveyorShortVarSpeed_SetSpeed**".

→ Another parameter must be added to the signal adapter in order to guarantee that the "**tsConveyorLong**" transport surface only moves when there is an active signal of one of the two associated speed controls. Select the parameter name "**active**" for this transport surface and give the new parameter the alias name "**paTsConveyorLong_SetActive**". Then, assign the following formula for this parameter, as indicated in Figure 12, step 1:

"((scConveyorLongConstSpeed_SetActive) | (scConveyorLongVarSpeed_SetActive))"

The character "**|**" stands for the OR function in this context. This ensures that the conveyor belt only moves when at least one speed control has been activated for it. Based on the logic in the prepared automation program, however, the transport surface can only be operated by one speed control at a time during regular operation.



A	Alias	Object	Object Type	Parameter	Value
✓	paTsConveyorLong_SetActive	tsConveyorLong	Transport Surface	active	true

Assign to	Formula
paTsConveyorLong_SetActive	((scConveyorLongConstSpeed_SetActive) | (scConveyorLongVarSpe

Formula: ((scConveyorLongConstSpeed_SetActive) | (scConveyorLongVarSpeed_SetActive))

Figure 12: Creating a parameter for a transport surface.

→ The same behavior as described previously for **"tsConveyorLong"** also applies to the **"tsConveyorShort"** transport surface. Select the "**active**" status for the new parameter named "**paTsConveyorShort_SetActive**".

Use the following formula for this: "**((scConveyorShortConstSpeed_SetActive)|(scConveyorShortVarSpeed_SetActive))**", in order to ensure the proper operation of the transport surface.

All the required parameters and signals are now defined in the signal adapter. Next, assign the name "**saSortingPlant**" for the signal adapter (see Figure 13, step 1). The prefix **"sa"** stands for **"signal adapter"**. Confirm the configuration of your new signal adapter by clicking the "**OK**" button (see Figure 13, step 2).

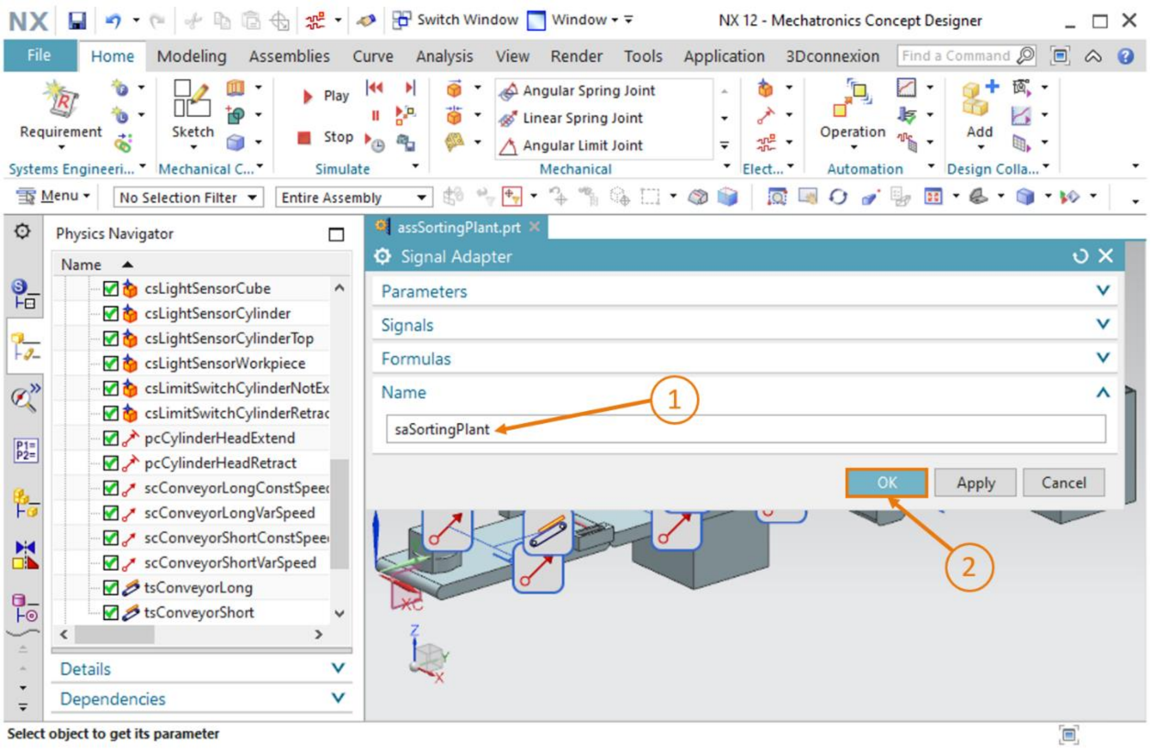


Figure 13: Creating the "saSortingPlant" signal adapter

A new window "**Add Symbols to Symbol Table**" now appears in which you are prompted to specify the symbol table in which the signals of your signal adapter are to be added as signals. Here, there is the option to add to an existing symbol table or to create a new symbol table. Because you have not yet created a symbol table in your project, click the "**New Symbol Table**" button (see Figure 14, step 1).

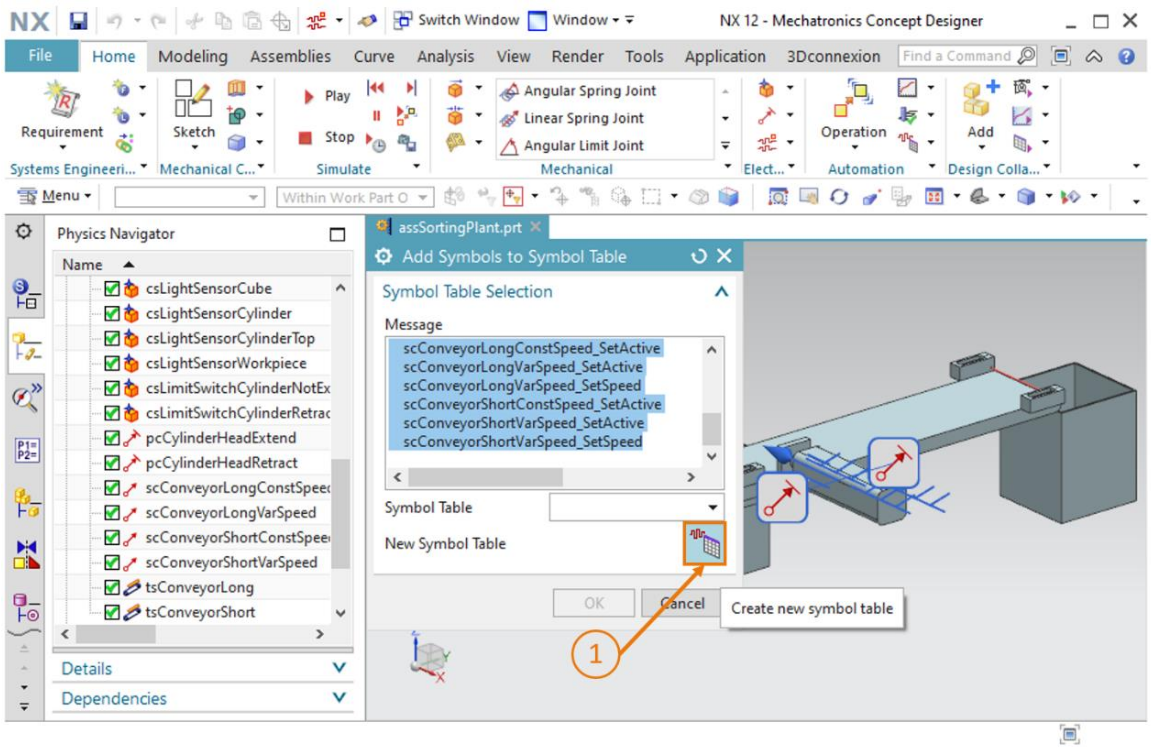

Add Symbols to Symbol Table
Symbol Table Selection
Message
scConveyorLongConstSpeed_SetActive
scConveyorLongVarSpeed_SetActive
scConveyorLongVarSpeed_SetSpeed
scConveyorShortConstSpeed_SetActive
scConveyorShortVarSpeed_SetActive
scConveyorShortVarSpeed_SetSpeed
Symbol Table
New Symbol Table
OK
Cancel
Create new symbol table
1
Physics Navigator
assSortingPlant.prt

Figure 14: Initiating the creation of a new symbol table for the signal adapter

The "**Symbol Table**" command window now appears. Here, you can define new symbols as well as assign a name for the symbol table. Because you are able to transfer the signals from your signal adapter in full, you do not need to define any new signals here. Assign the name "**stSortingPlant**" to the symbol table (see Figure 15, step 1) and click on the "**OK**" button (see Figure 15, step 2). The prefix **"st"** stands for **"signal table"**.

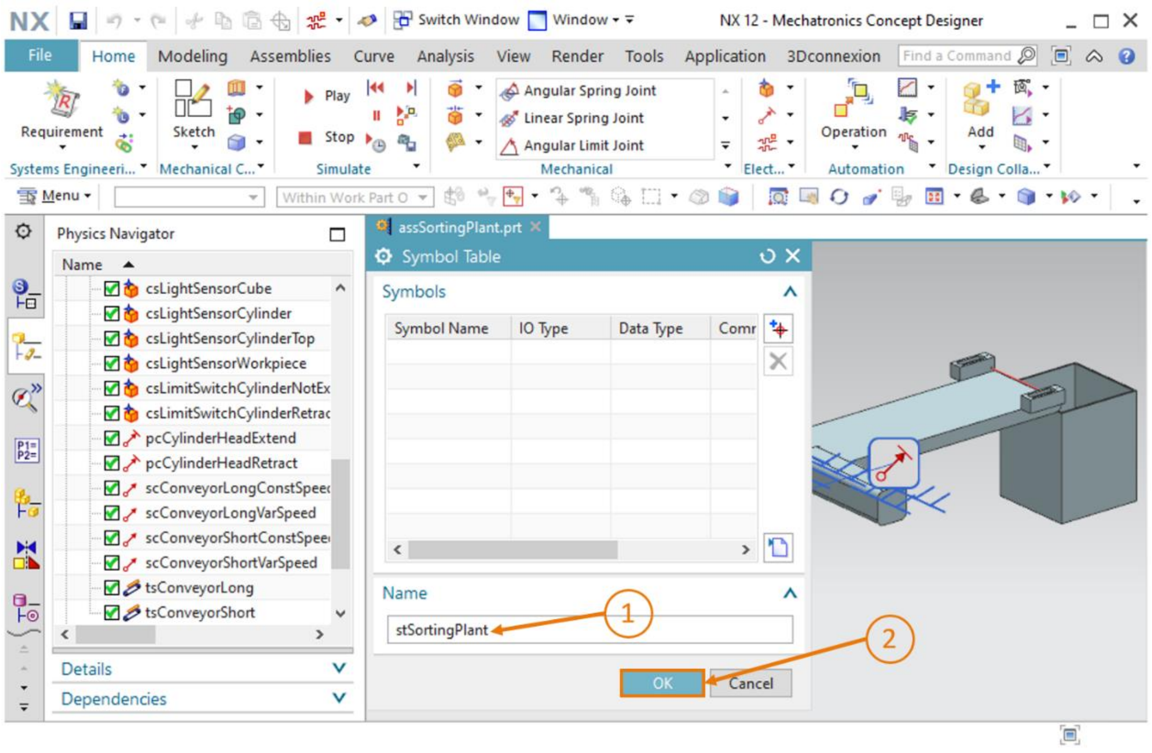
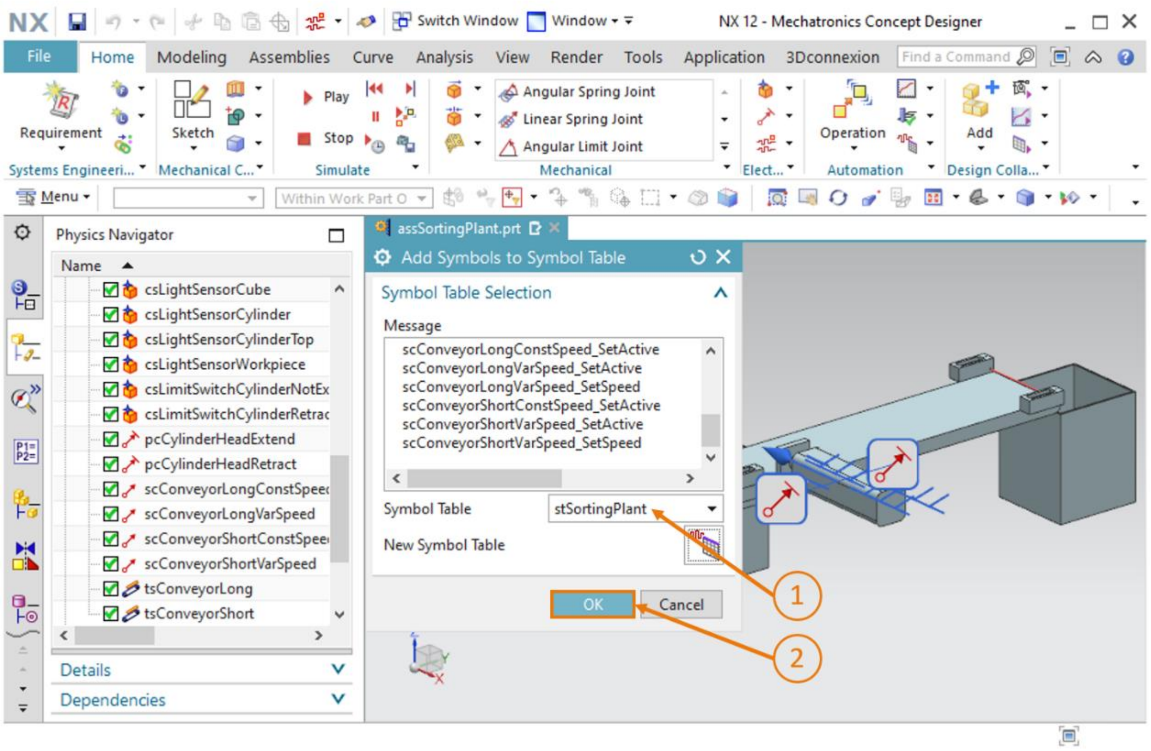


Figure 15: Completing the creation of a new symbol table for the signal adapter

You are now returned to the "**Add Symbols to Symbol Table**" window. Here, if not already selected, you should select the "**stSortingPlant**" symbol table you just created, as shown in Figure 16, step 1. Finish the creation process by clicking the "**OK**" button (see Figure 16, step 2).



Message:

```
scConveyorLongConstSpeed_SetActive
scConveyorLongVarSpeed_SetActive
scConveyorLongVarSpeed_SetSpeed
scConveyorShortConstSpeed_SetActive
scConveyorShortVarSpeed_SetActive
scConveyorShortVarSpeed_SetSpeed
```

Symbol Table: stSortingPlant

Figure 16: Completing the symbol assignment for the signal adapter

You have now inserted all the required signals in your dynamic 3D model, and you can proceed to establishing a signal connection to a virtual PLC. But first save the changes to the model by clicking the "**Save**" button .

7.2 Creating a signal connection between a virtual PLC and a digital twin

To create a signal connection, a virtual PLC must already be in operation. In this section, you must therefore return to the TIA Portal and PLCSIM Advanced. Proceed as follows to create the connection:

→ Extract the archive provided with this module in your operating system (see Chapter 7) and save the content of the "**fullPlcBasic**" folder to a folder of your choice. The folder contains the automation program already used in Module 1 and described in Module 2.

→ Open the **TIA Portal** and retrieve the "**150-006_DigitalTwinAtEducation_TIAP_Basic.zap15**" project from the folder you just created. Use the procedure dekinoxscribed in **Chapter 7.1 of Module 1 of the DigitalTwin@Education workshop series** for this.

→ Compile both the hardware configuration and the software of the automation program. Follow the explanations in **Chapter 7.2 in Module 1 of this workshop series** for this.

→ Open the "**S7-PLCSIM Advanced**" program and start a new instance of a virtual PLC. Name this instance "**DigTwinAtEdu_PLCSIM**". Then download your automation program to the virtual PLC and wait until the CPU status changes to "**Start**", i.e. a green box appears in front of the instance name. Proceed here according to the descriptions in **Chapter 7.3 of Module 1 of this workshop series**.

The virtual PLC is now ready for operation and you can now configure the signal connection to the dynamic 3D model. Go back to your dynamic 3D model with signals in the Mechatronics Concept Designer and follow the steps below:

→ Select the "**Signal Mapping**" command in the "**Automation**" menu group (see Figure 17, step 1). The "**Signal Mapping**" dialog window opens. There, you must first select the external signal source. To do this, go to the **"External Signal Type"** group and select "**PLCSIM Adv**" as the type, because your aim is to connect to PLCSIM Advanced (see Figure 17, step 2). At this point, your dynamic model does not know the instance in PLCSIM Advanced to which a connection is to be established. For this reason, click the "**Settings**" button for the "**PLCSIM Adv Instances**" item (see Figure 17, step 3).

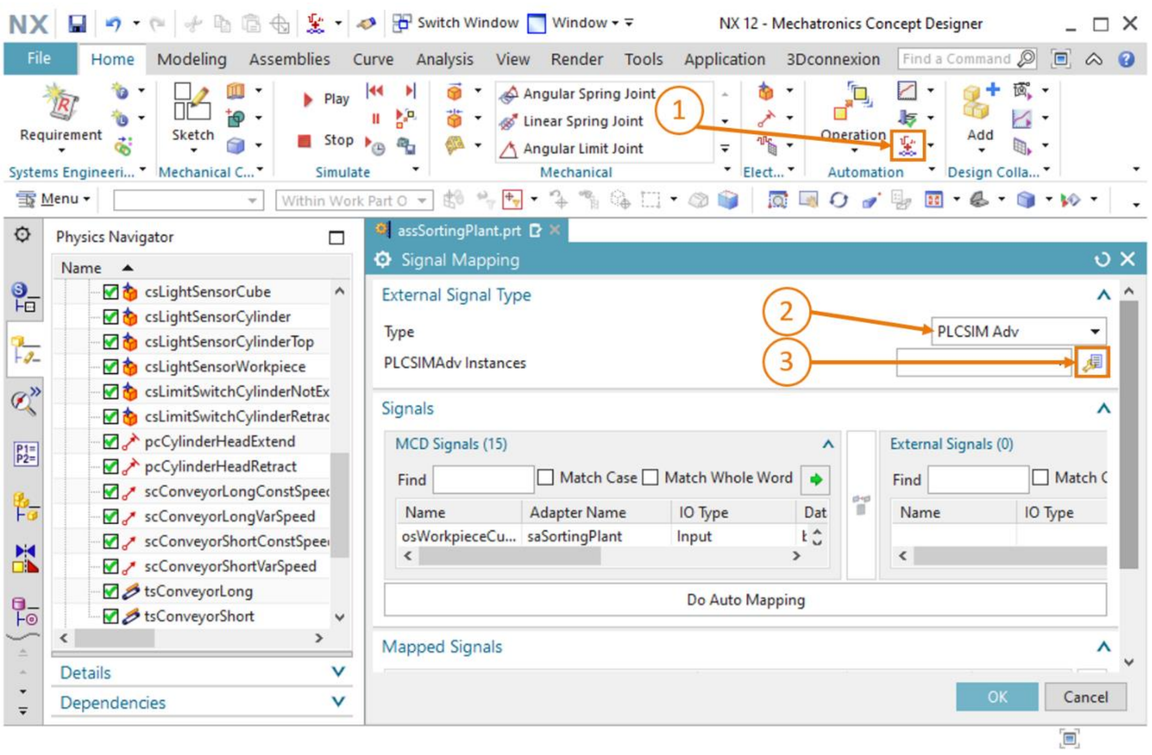
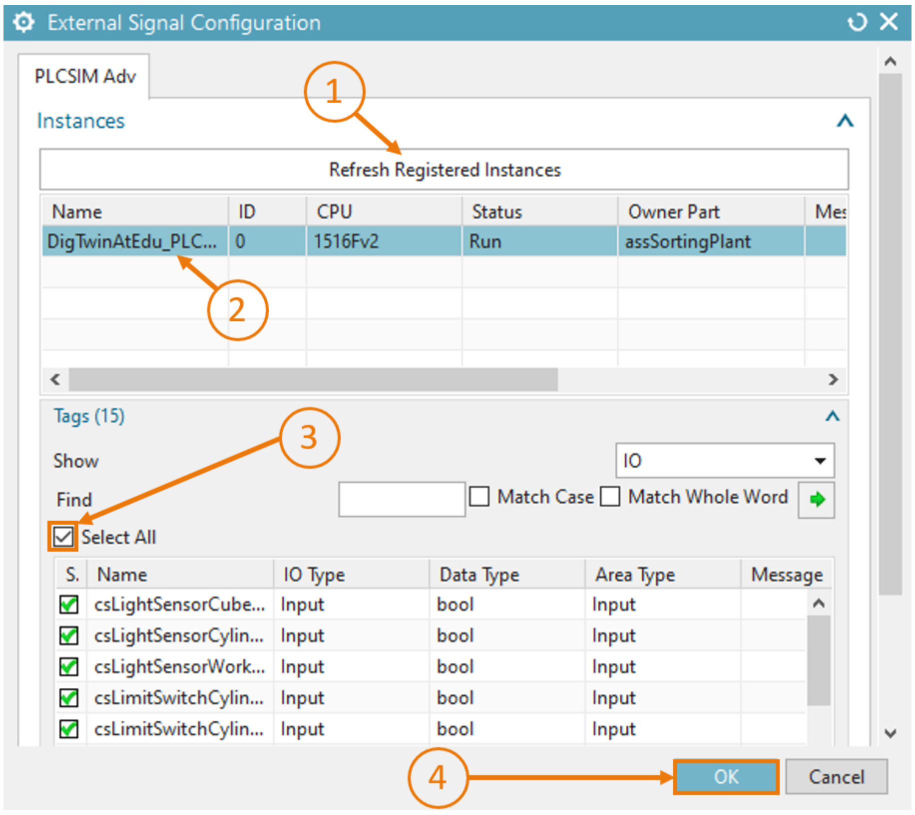


Figure 17: Selection of signal mapping via PLCSIM Advanced

NOTE

If signal connections have already been created for the dynamic model, the selection bar of the **"PLCSIMAdv Instances"** item shows **all the PLCSIM Advanced instances that have ever been used previously for this model.** Note, however, that these instances do not necessarily exist and are not necessarily valid. To check, click on the **"Settings"** button and look at the current status of the respective instance.

→ The "**External Signal Configuration**" window now opens. Here, you can select the desired instance and enable the associated tags for the signal mapping. Next, click the "**Refresh Registered Instances**" button (see Figure 18, step 1). Your previously started and downloaded virtual PLC instance appears. The **"Run"** status indicates that this virtual PLC is accessible. After selecting this instance, as shown in Figure 18, step 2, you see the I/O signals of the automation program. Insert all the available tags by selecting the "**Select All**" check box (see Figure 18, step 3). Confirm the selection by clicking "**OK**" (see Figure 18, step 4).



Name	ID	CPU	Status	Owner Part	Mes
DigTwinAtEdu_PLC...	0	1516Fv2	Run	assSortingPlant	

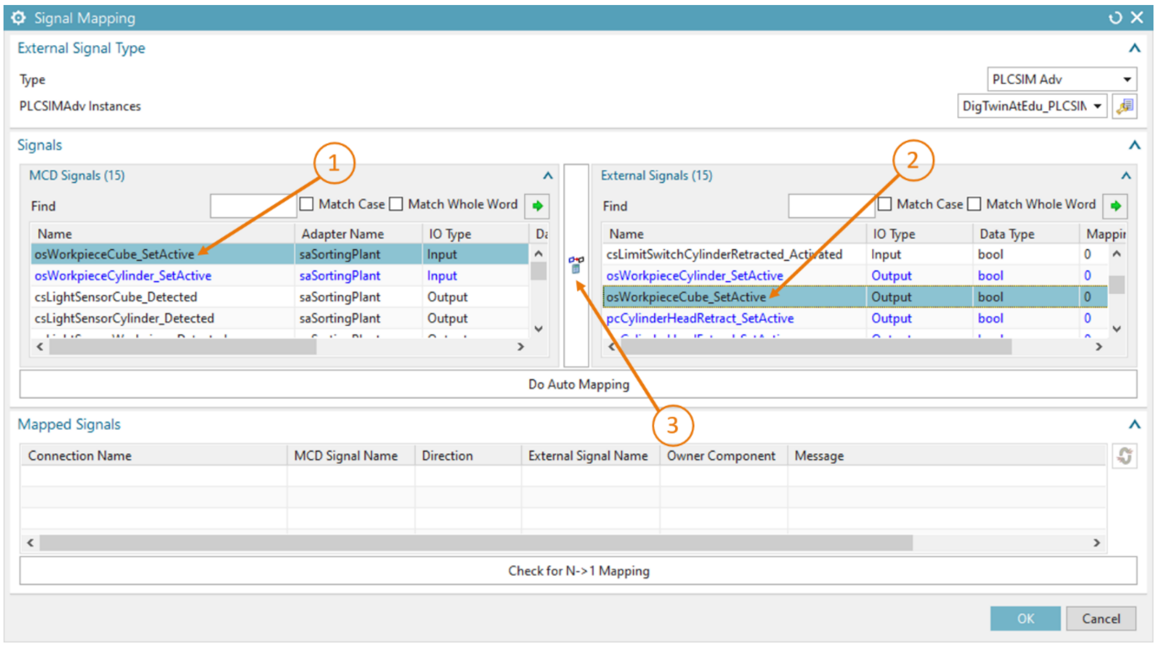
S.	Name	IO Type	Data Type	Area Type	Message
✓	csLightSensorCube...	Input	bool	Input	
✓	csLightSensorCylin...	Input	bool	Input	
✓	csLightSensorWork...	Input	bool	Input	
✓	csLimitSwitchCylin...	Input	bool	Input	
✓	csLimitSwitchCylin...	Input	bool	Input	

Figure 18: Enabling tags from PLCSIM Advanced instance for signal mapping

NOTE

Whenever changes or additions are made to the automation program, the registered instance for the signal mapping must be updated and, if necessary, additional signals must be added.

→ You are returned to the "**Signal Mapping**" command window. There, you will find the virtual PLC you have just selected and the external signals available with it in the right part of the window. You can now start mapping the signals. First, select the "**osWorkpiece Cube_SetActive**" signal in the "**MCD Signals**" table in the left part of the window (see Figure 19, step 1). Then search for the corresponding signal from the automation program in the "**External Signals**" table. For obvious reasons, identical names from the two programs were selected here, as illustrated in Figure 19, step 2. Next, click the "**Map Signal**" button to establish a connection between the two signals (see Figure 19, step 3). It is important to mention here that input signals in MCD can only be connected to output signals of a PLC, and vice versa.



Type: PLCSIM Adv
PLCSIMAdv Instances: DigTwinAtEdu_PLCSIN

MCD Signals (15)

Name	Adapter Name	IO Type	Da
osWorkpieceCube_SetActive	saSortingPlant	Input	
osWorkpieceCylinder_SetActive	saSortingPlant	Input	
csLightSensorCube_Detected	saSortingPlant	Output	
csLightSensorCylinder_Detected	saSortingPlant	Output	

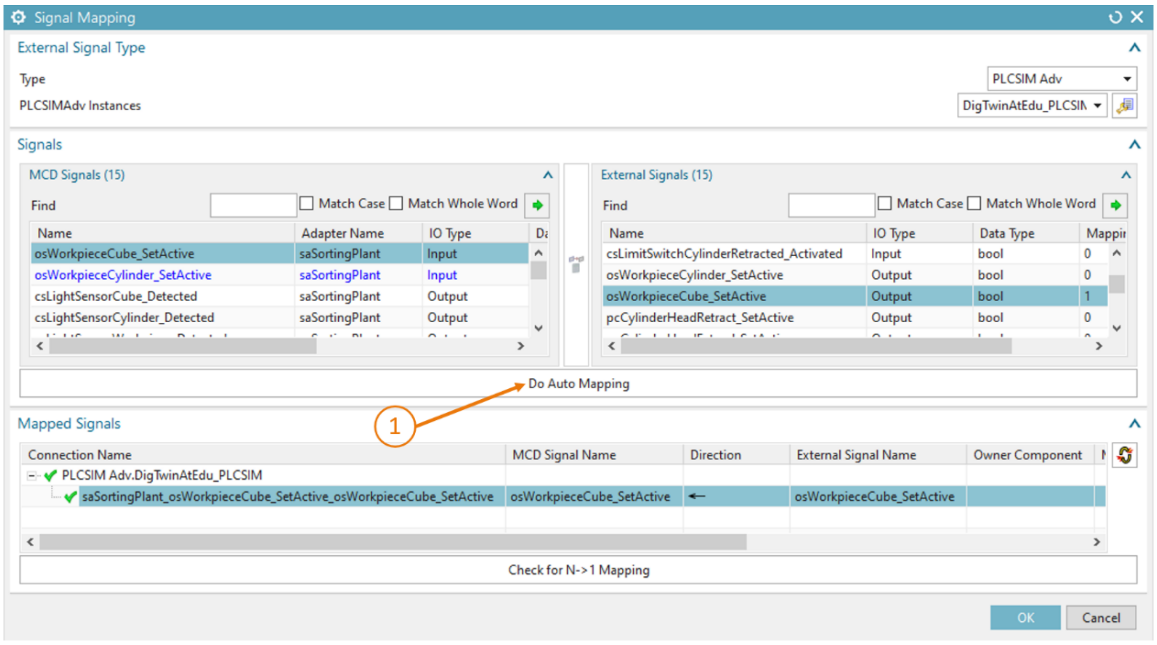
External Signals (15)

Name	IO Type	Data Type	Mappi
csLimitSwitchCylinderRetracted_Activated	Input	bool	0
osWorkpieceCylinder_SetActive	Output	bool	0
osWorkpieceCube_SetActive	Output	bool	0
pcCylinderHeadRetract_SetActive	Output	bool	0

Connection Name	MCD Signal Name	Direction	External Signal Name	Owner Component	Message

Figure 19: Mapping an MCD signal to an external signal

→ You can now see the signal mapping you just completed in the table in the "**Mapped Signals**" group. Now insert the other mappings. Because the names of the MCD signals match the tag names of the automation program in this model, you can select the "**Do Auto Mapping**" button to have the program perform this process automatically (see Figure 20, step 1).


Signal Mapping

External Signal Type

Type: PLCSIM Adv

PLCSIMAdv Instances: DigTwinAtEdu_PLCSIM

Signals

MCD Signals (15)

Find ☐ Match Case ☐ Match Whole Word

Name	Adapter Name	IO Type
osWorkpieceCube_SetActive	saSortingPlant	Input
osWorkpieceCylinder_SetActive	saSortingPlant	Input
csLightSensorCube_Detected	saSortingPlant	Output
csLightSensorCylinder_Detected	saSortingPlant	Output

External Signals (15)

Find ☐ Match Case ☐ Match Whole Word

Name	IO Type	Data Type	Mappi
csLimitSwitchCylinderRetracted_Activated	Input	bool	0
osWorkpieceCylinder_SetActive	Output	bool	0
osWorkpieceCube_SetActive	Output	bool	1
pcCylinderHeadRetract_SetActive	Output	bool	0

Do Auto Mapping

Mapped Signals

Connection Name	MCD Signal Name	Direction	External Signal Name	Owner Component
✓ PLCSIM Adv.DigTwinAtEdu_PLCSIM				
✓ saSortingPlant_osWorkpieceCube_SetActive_osWorkpieceCube_SetActive	osWorkpieceCube_SetActive	←	osWorkpieceCube_SetActive	

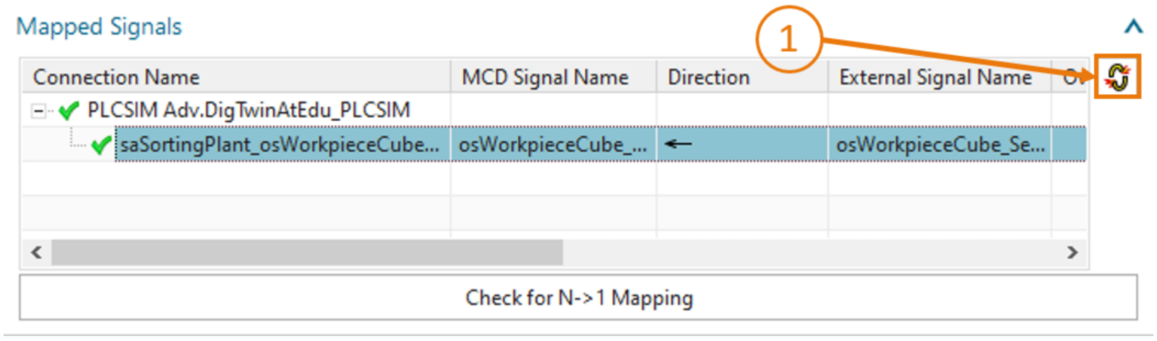
Check for N->1 Mapping

OK Cancel

Figure 20: Connecting all signals using automatic mapping

NOTE

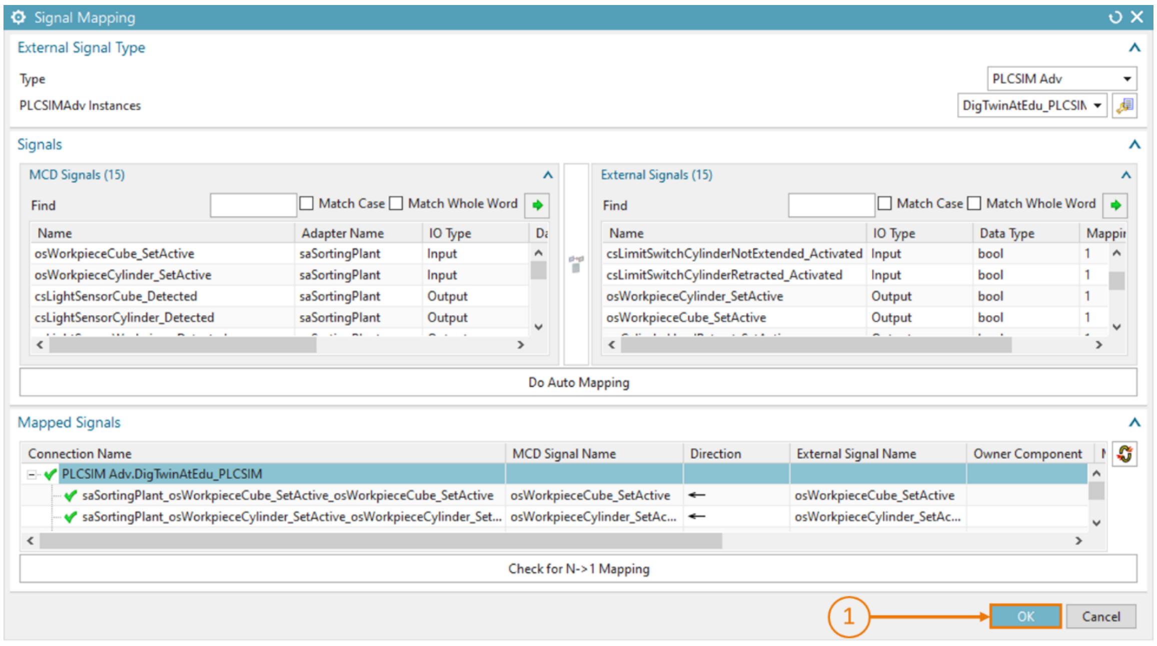
If you have incorrectly mapped a signal, you can select the corresponding mapping in the table of the "Mapped Signals" command group, and disconnect it again by clicking the "**Break**" button (see Figure 21, step 1). You must then create the correct mapping.



Connection Name	MCD Signal Name	Direction	External Signal Name	Ov
PLCSIM Adv.DigTwinAtEdu_PLCSIM				
saSortingPlant_osWorkpieceCube...	osWorkpieceCube_...	←	osWorkpieceCube_Se...	

Figure 21: Breaking signal mapping

→ The automatic mapping process has now interconnected all 15 signals between the dynamic 3D model and the virtual PLC. Check the mappings for correctness and complete the signal mapping by clicking the "**OK**" button (see Figure 22, step 1).



Type: PLCSIM Adv

PLCSIMAdv Instances: DigTwinAtEdu_PLCSIN

MCD Signals (15)

Name	Adapter Name	IO Type	Da
osWorkpieceCube_SetActive	saSortingPlant	Input	
osWorkpieceCylinder_SetActive	saSortingPlant	Input	
csLightSensorCube_Detected	saSortingPlant	Output	
csLightSensorCylinder_Detected	saSortingPlant	Output	

External Signals (15)

Name	IO Type	Data Type	Mappir
csLimitSwitchCylinderNotExtended_Activated	Input	bool	1
csLimitSwitchCylinderRetracted_Activated	Input	bool	1
osWorkpieceCylinder_SetActive	Output	bool	1
osWorkpieceCube_SetActive	Output	bool	1

Mapped Signals

Connection Name	MCD Signal Name	Direction	External Signal Name	Owner Component
PLCSIM Adv.DigTwinAtEdu_PLCSIM				
saSortingPlant_osWorkpieceCube_SetActive_osWorkpieceCube_SetActive	osWorkpieceCube_SetActive	←	osWorkpieceCube_SetActive	
saSortingPlant_osWorkpieceCylinder_SetActive_osWorkpieceCylinder_Set...	osWorkpieceCylinder_SetAc...	←	osWorkpieceCylinder_SetAc...	

Figure 22: Confirming signal mapping between dynamic model and virtual PLC

The connection between the dynamic 3D model in NX/MCD and the automation program in your virtual PLC is now established. Save your model by clicking the "**Save**" button .

7.3 Testing the digital twin with the virtual PLC

In this chapter, you are to operate your digital twin in interaction with an automation program in a virtual PLC and validate the functionality. To do this, follow the procedure below:

→ Having already downloaded the automation program into an instance of a virtual PLC in Chapter 7.2, you now start the HMI using the "**WinCC Runtime Advanced**" simulation tool. This is to be done using the TIA Portal. Use the procedure in **Chapter 7.2 of Module 1 of the DigitalTwin@Education workshop series** for this.

→ Then go to the **"Mechatronics Concept Designer"** program and start a simulation for your digital twin. Run the "**Play**" command ▶ in the "**Simulation**" menu group for this.

→ Perform the **two test scenarios from the first module of this workshop series** for your digital twin and validate the functionality of your digital twin. Follow the instructions in **Chapter 7.6 of Module 1 of this workshop series** for this. You can see that the digital twin you created yourself by completing Modules 4 - 6 of this workshop series behaves the same as the prepared model you used for the first three modules. At the end of your test series, stop the simulation in MCD, close the simulated HMI instance and exit your virtual PLC.

Of course, you are free to check your digital twin with your optimized automation program from **Module 3**.

You have now arrived at the end of this training module. With the knowledge you have acquired, you can independently create your own digital twins and perform virtual commissioning for your automation project.

8 Checklist – Step-by-step instructions

The following checklist helps students/trainees to independently check whether all steps of the step-by-step instructions have been carefully completed and enables them to successfully complete the module on their own.

No.	Description	Checked
1	Your dynamic model from Module 5 was successfully expanded to include the required signals.	
2	A valid signal connection was created between your digital twin and the virtual PLC.	
3	The digital twin you created on your own was able to be completely and successfully validated by simulating the test scenarios from Module 1 of this workshop series.	

Table 1: Checklist for "Signal Creation for a Dynamic 3D Model in the Mechatronics Concept Designer CAE System"

9 Additional information

You can find additional information as an orientation aid for initial and advanced training, for example: Getting Started, videos, tutorials, apps, manuals, programming guidelines and trial software/firmware, at the following link:

Preview "Additional information" – In preparation

Here are some interesting links:

[1] support.industry.siemens.com/cs/document/90885040/programming-guideline-for-s7-1200-s7-1500?dti=0&lc=en-US

[2] support.industry.siemens.com/cs/document/109756737/guide-to-standardization?dti=0&lc=en-US

[3] omg.org/spec/UML/2.5.1/PDF

[4] geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/

[5] geeksforgeeks.org/unified-modeling-language-uml-state-diagrams/

Further Information

Siemens Automation Cooperates with Education
siemens.com/sce

SCE Learn-/Training Documents
siemens.com/sce/documents

SCE Trainer Packages
siemens.com/sce/tp

SCE Contact Partners
siemens.com/sce/contact

Digital Enterprise
siemens.com/digital-enterprise

Totally Integrated Automation (TIA)
siemens.com/tia

TIA Portal
siemens.com/tia-portal

TIA Selection Tool
siemens.com/tia/tia-selection-tool

SIMATIC Controller
siemens.com/controller

SIMATIC Technical Documentation
siemens.com/simatic-docu

Industry Online Support
support.industry.siemens.com

Product catalogue and online ordering system Industry Mall
mall.industry.siemens.com

Siemens
Digital Industries, FA
P.O. Box 4848
90026 Nuremberg
Germany

siemens.com/sce