



SIEMENS

Application Example • 12/2016

SIMATIC S7-PLCSIM Advanced: Co-Simulation via API

STEP 7 V14, S7-PLCSIM Advanced V1.0



<https://support.industry.siemens.com/cs/ww/en/view/109739660>

Warranty and Liability

Note

The Application Examples are not binding and do not claim to be complete regarding the circuits shown, equipping and any eventuality. The Application Examples do not represent customer-specific solutions. They are only intended to provide support for typical applications. You are responsible for ensuring that the described products are used correctly. These Application Examples do not relieve you of the responsibility to use safe practices in application, installation, operation and maintenance. When using these Application Examples, you recognize that we cannot be made liable for any damage/claims beyond the liability clause described. We reserve the right to make changes to these Application Examples at any time without prior notice.

If there are any deviations between the recommendations provided in these Application Examples and other Siemens publications – e.g. Catalogs – the contents of the other documents have priority.

We do not accept any liability for the information contained in this document. Any claims against us – based on whatever legal reason – resulting from the use of the examples, information, programs, engineering and performance data etc., described in this Application Example shall be excluded. Such an exclusion shall not apply in the case of mandatory liability, e.g. under the German Product Liability Act (“Produkthaftungsgesetz”), in case of intent, gross negligence, or injury of life, body or health, guarantee for the quality of a product, fraudulent concealment of a deficiency or breach of a condition which goes to the root of the contract (“wesentliche Vertragspflichten”). The damages for a breach of a substantial contractual obligation are, however, limited to the foreseeable damage, typical for the type of contract, except in the event of intent or gross negligence or injury to life, body or health. The above provisions do not imply a change of the burden of proof to your detriment.

Any form of duplication or distribution of these Application Examples or excerpts hereof is prohibited without the expressed consent of the Siemens AG.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks. In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions only form one element of such a concept.

Customer is responsible to prevent unauthorized access to its plants, systems, machines and networks. Systems, machines and components should only be connected to the enterprise network or the internet if and to the extent necessary and with appropriate security measures (e.g. use of firewalls and network segmentation) in place.

Additionally, Siemens' guidance on appropriate security measures should be taken into account. For more information about industrial security, please visit <http://www.siemens.com/industrialsecurity>.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends to apply product updates as soon as available and to always use the latest product versions. Use of product versions that are no longer supported, and failure to apply latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under <http://www.siemens.com/industrialsecurity>.

Table of Contents

Warranty and Liability	2
1 Introduction	4
1.1 Overview.....	4
1.2 Mode of Operation.....	5
1.2.1 Overview.....	5
1.2.2 Range of functions	8
1.3 Components used	9
2 Engineering	10
2.1 Configuration in the TIA Portal	10
2.2 Programming in Visual Studio.....	11
2.2.1 Integrating the API library.....	11
2.2.2 Overview user interface.....	12
2.2.3 Programming the PLC instance	13
2.2.4 Programming the co-simulation	19
2.3 Operating the Application Example	21
2.3.1 Switching on virtual controller	21
2.3.2 Loading TIA Portal project into the virtual controller	23
2.3.3 Starting WinCC Runtime	26
2.3.4 Starting the co-simulation.....	27
2.3.5 Restarting the conveyor system.....	28
2.3.6 Simulating an error	29
2.3.7 Removing and acknowledging error.....	30
2.3.8 Stopping the co-simulation	31
2.3.9 Switching off virtual controller	32
2.3.10 Closing application	32
2.3.11 Error handling.....	33
3 Valuable Information	33
3.1 Basics of S7-PLCSIM Advanced.....	33
3.1.1 Installation	33
3.1.2 Difference between S7-PLCSIM V14 and PLCSIM Advanced V1.0	34
3.1.3 User interface	35
3.1.4 General properties.....	37
3.1.5 Simulation.....	38
3.1.6 User interfaces (API)	41
3.2 Details on the functionality	43
3.2.1 TIA Portal program	43
3.2.2 Co-simulation program	45
4 Appendix	46
4.1 Service and Support.....	46
4.2 Links and Literature	47
4.3 Change documentation	47

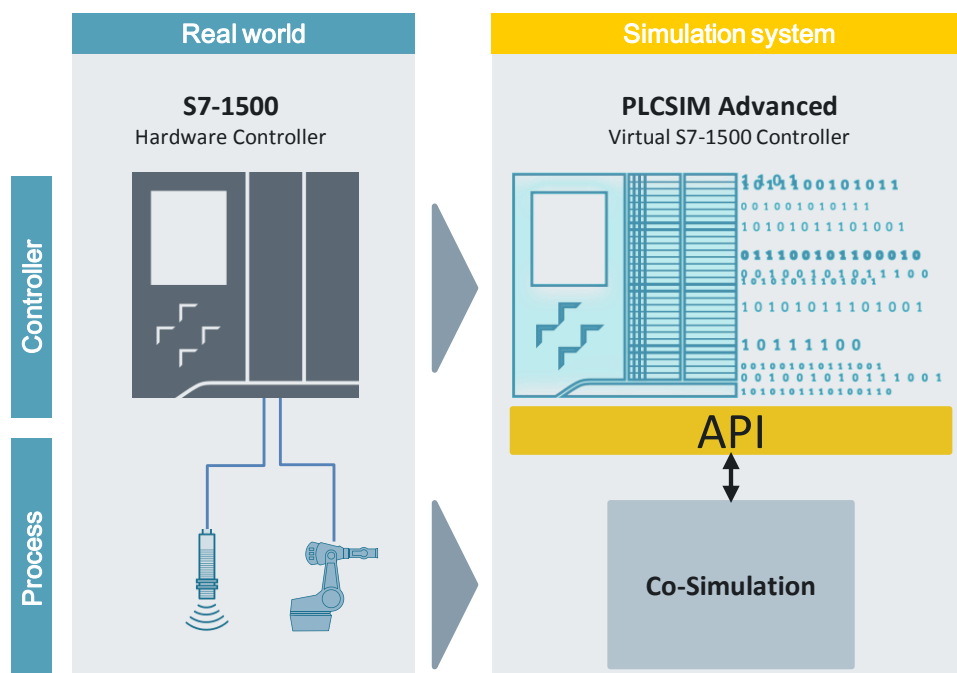
1 Introduction

1.1 Overview

With the SIMATIC S7-PLCSIM Advanced virtual controllers for the simulation of a S7-1500 or ET 200SP CPU are created and used for the comprehensive simulation of functions. Therefore no real controllers are required to test a STEP 7 program.

The virtual controllers can also be tested and validated in the context of a system or machine. The user interface (API) is used to connect the virtual controller to a plant/system or machine simulation (co-simulation).

Figure 1-1: Overview



Advantages of the application example

- Introduction to the use of API.
- C# example code on which you can establish your own applications.

Topics not covered by this application

This application example does not contain a description of the following topics:

- Basics of object-oriented programming
- Basics of programming environment, for example, Microsoft Visual Studio
- Basics of TIA Portal configuration

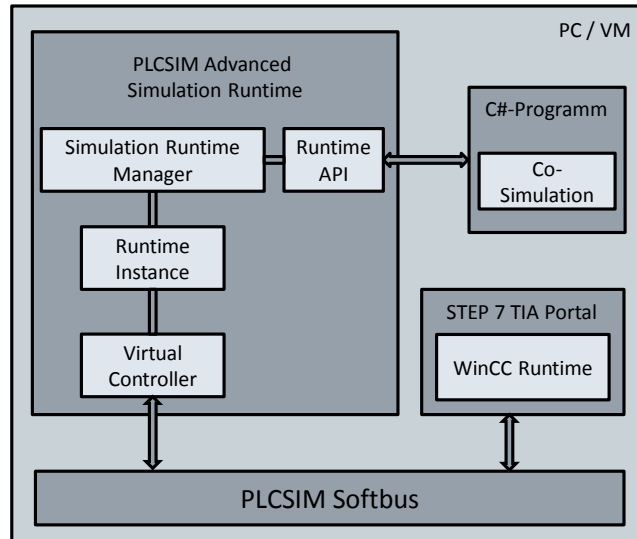
Sufficient knowledge of these topics is assumed.

1.2 Mode of Operation

1.2.1 Overview

The following figure presents an overview of the functional principle of the application example.

Figure 1-2: Overview of the systems



PLCSIM Advanced

With S7-PLCSIM Advanced as independent simulation system, the created user program can be simulated and tested on a virtual controller.

The Runtime API of S7-PLCSIM Advanced (details see chapter [3.1.6](#)) provides user interfaces for access to the simulation runtime. These user interfaces offer the following options via a program (for example, C# program):

- Creation of a runtime instance of a virtual controller.
- Changing the mode of the virtual controller.
- Exchanging I/O data with a co-simulation.

In this application example a local communication via the PLCSIM Softbus is used. However, with S7-PLCSIM Advanced a distributed communication via a virtual Ethernet adapter is also possible.

STEP 7 (TIA Portal) program

A STEP 7 program created in STEP 7 V14 (details see chapter 3.2.1) controls a conveyor system. For a comprehensive function test, the STEP 7 program is loaded via S7-PLCSIM Advanced into a virtual S7-1500 controller. This controller interacts via the API with a co-simulation (system simulation), in order to validate the STEP 7 program in the context of the system.

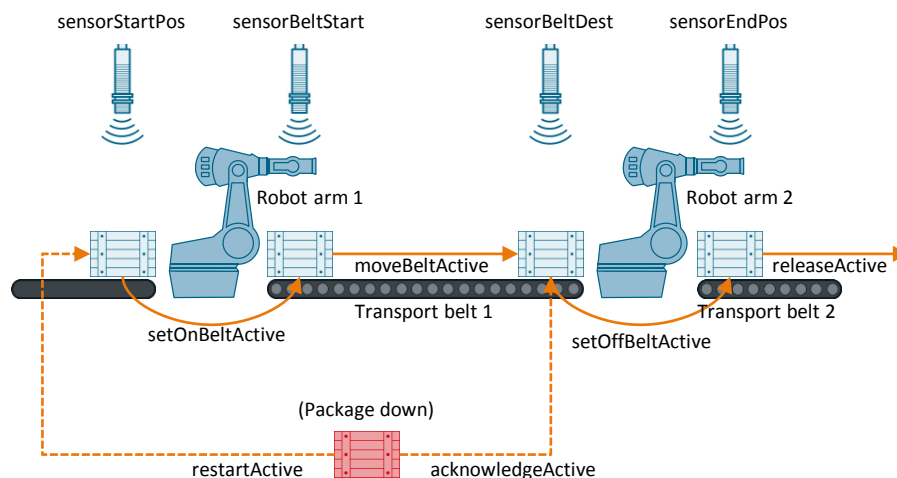
The material handling system to be controlled consists of two transport belts and two robot arms. The following input and output tags are defined so that the controller communicates with the material handling system:

Table 1-1: Input and output tags

Tags	Data type	Type	Description
sensorStartPos	BOOL	Input	Sensor signal: Starting position
sensorBeltStartPos	BOOL	Input	Sensor signal: Starting position transport belt 1
sensorBeltDest	BOOL	Input	Sensor signal: Target position transport belt 1
sensorEndPos	BOOL	Input	Sensor signal: End position
setOnBeltActive	BOOL	Output	Control signal: Enable robot arm 1
moveBeltActive	BOOL	Output	Control signal: Enable transport belt 1
setOffBeltActive	BOOL	Output	Control signal: Enable robot arm 2
releaseActive	BOOL	Output	Control signal: Enable transport belt 2
acknowledgeActive	BOOL	Output	Control signal: Acknowledge error
restartActive	BOOL	Output	Control signal: Restart system

The according control signal is activated based on the active sensor signals. For transport belt 1 (moveBeltActive) a monitoring time of 5 s is configured. If the package does not reach the target position within this time, an error for a fallen down package is shown ("Package down"). You can then restart the system or acknowledge the error when you put the package back on the transport belt 1.

Figure 1-3: Overview of the conveyor system



The visualization of the system is simulated with WinCC Runtime. The system can be monitored and controlled in WinCC Runtime via an HMI screen:

- Restart (restartActive)
- Acknowledge error (acknowledgeActive)

C# program

In the external application (C# program) the following is programmed:

- Functions of the user interface (Runtime API) in order to interact with the virtual controller of S7-PLCSIM Advanced.
- System simulation of the conveyor system (co-simulation).

The co-simulation reacts to the control signals (outputs) of the virtual controller and simulates all required sensor signals (inputs) for the simulation of the sequential control (details chapter 3.2.2).

For the exchange of the I/O data the Runtime API writes and reads from the memory area that is synchronized on the cycle control point with the internal process image of the virtual S7-1500 controller.

Table 1-2: Data exchange

Virtual Controller	Data exchange	Co-Simulation
setOnBeltActive	→	setOnBeltActive
moveBeltActive	→	moveBeltActive
setOffBeltActive	→	setOffBeltActive
releaseActive	→	releaseActive
acknowledgeActive	→	acknowledgeActive
restartActive	→	restartActive
sensorStartPos	←	sensorStartPos
sensorBeltStart	←	sensorBeltStart
sensorBeltDest	←	sensorBeltDest
sensorEndPos	←	sensorEndPos

With the co-simulation you can simulate or remove errors, for example, drop a package or place a dropped package back on transport belt 1.

Note

The machines of the conveyor system are simulated in a non-realistic way in the co-simulation. The movements are only simulated by a time constant.

1.2.2 Range of functions

In this application example the following functions are used and shown in Runtime API.

Table 1-3: API functions

Function	Description	See
<code>RegisterInstance()</code>	Registers a new instance of a virtual controller in the Runtime Manager.	P. 13
<code>UnregisterInstance()</code>	Unregisters an instance from the Runtime Manager.	P. 13
<code>PowerOn()</code>	Creates the process for the simulation runtime instance and starts the firmware of the virtual controller.	P. 14
<code>PowerOff()</code>	Shuts down the simulation runtime instance and closes its process.	P. 15
<code>Run()</code>	Calls on the virtual controller to change to RUN operating state.	P. 15
<code>Stop()</code>	Calls on the virtual controller to change to STOP operating state.	P. 16
<code>SetIPSuite()</code>	Sets the IP suite of the network interface of a virtual controller.	P. 14
<code>UpdateTagList()</code>	Reads the tags from the virtual controller and writes them to the shared storage arranged by name.	P. 18
<code>ReadBool()</code>	Reads the value of a PLC tag symbolically.	P. 18
<code>WriteBool()</code>	Writes the value of a PLC tag symbolically.	P. 18
<code>IsAlwaysSendOnEndOfCycleEnabled{get; set;}</code>	Provides or sets the mode <code>AlwaysSendOnEndOfCycle</code> . When the mode is set, the event <code>OnEndOfCycle</code> is triggered for each mode after each end of cycle.	P. 13
<code>CommunicationInterface{get; set;}</code>	Sets the communication interface of the virtual controller or returns it: Local communication (softbus) or TCP/IP.	P. 13
<code>OnConfigurationChanged</code>	This event is triggered when the configuration of the virtual controller has changed.	P. 13
<code>OnEndOfCycle</code>	This event is triggered when the virtual controller has reached the end of the main cycle.	P. 13

1.3 Components used

Hardware

Note

No real hardware is used in this application example. All hardware components configured are simulated.

Table 1-4: Hardware components

Component	Number	Article number	Note
CPU 1516-3 PN/DP	1	6ES7516-3AN01-0AB0	V2.0
TP900 Comfort	1	6AV2124-0JC01-0AX0	V14.0.0.0

Software

Table 1-5: Software components

Component	Number	Article number
STEP 7 Professional V14	1	6ES7822-1AE04-0YA5
WinCC Advanced V14	1	6AV2102-0AA04-0AH5
S7-PLCSIM Advanced V1.0	1	6ES7823-1FE00-0YA5
Microsoft Visual Studio Ultimate 2013	1	-

Downloads

Table 1-6: Downloads

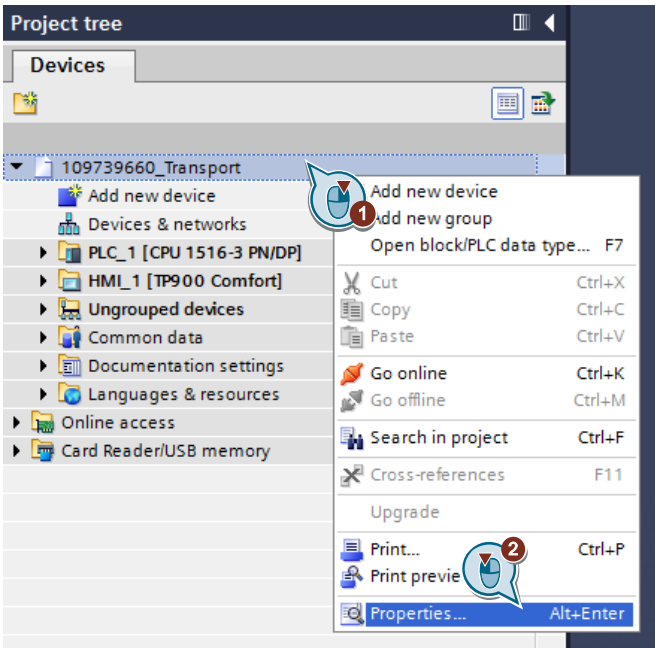
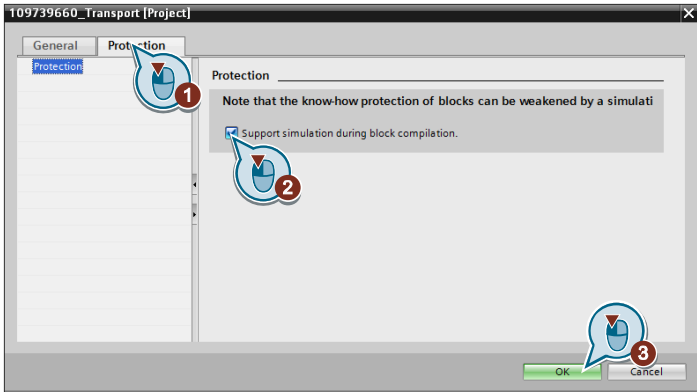
File	Content
109739660_PLCSIM_Advanced_DOC_V10_en.pdf	This document
109739660_PLCSIM_Advanced_S7_PROJ_V10.zip	TIA Portal project (system control)
109739660_PLCSIM_Advanced_COSIM_APPL_V10.zip	System simulation (application and Visual Studio Project)

2 Engineering

2.1 Configuration in the TIA Portal

The enclosed TIA Portal project does not require any further configuration. If you create your own project, make the following settings in order to simulate with S7-PLCSIM Advanced.

Table 2-1: Setting in TIA Portal

No.	Action
1.	<p>1. Open the context menu for the project by right clicking it. 2. Click on "Properties".</p> 
2.	<p>1. Select the "Protection" tab. 2. Enable the "Support simulation during block compilation" option box. 3. Confirm with "OK".</p> 

2.2 Programming in Visual Studio

The Visual Studio project included facilitates your access to programming a S7-PLCSIM Advanced application with a co-simulation.

In the Visual Studio project some basic functions are already programmed, for example, switching on virtual controller and exchanging I/O data. With this Visual Studio project you can develop your applications further.

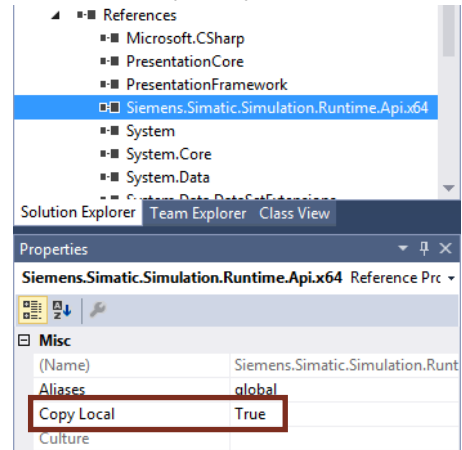
In chapter 2.2.2 to 2.2.4 you find an overview of the C#- and API functions that are programmed in this application example.

2.2.1 Integrating the API library

Note

For the path to be updated in the included project you have to delete the existing reference to the library “Siemens.Simatic.Simulation.Runtime.Api.x64” and integrate it again.

Table 2-2: Integrating the API library

No.	Action
1.	Create a reference in the project for the S7-PLCSIM Advanced library. It is located in the following directory in “...\Program Files (x86)\Common Files\Siemens\PLCSIMADV\API”.
2.	Set the property “Copy Local” of the reference to “True”. 

2.2.2 Overview user interface

In graphical user interface of the external application, commands (1) are integrated to interact with the virtual controller (PLC Instance) and the co-simulation.

The list box (2) shows info and error messages. The text field (3) shows the respective status.

Figure 2-1: GUI

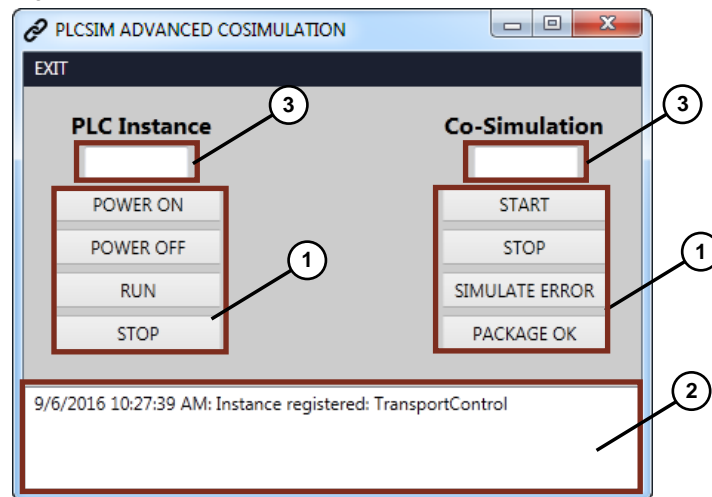
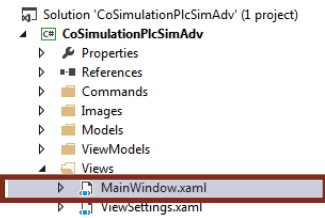


Table 2-3: Commands

No.	Description
1.	Open the project "CoSimulationPlcSimAdv" from the included download "109739660_PLCSIM_Advanced_COSIM_APPL_V10.zip".
2.	<p>Open the view of the main window "MainWindow.xaml" in the project folder explorer in the "Views" folder.</p>  <p>Here, in the XAML code amongst others you will find the integrated commands for the buttons or the integrated data for the display fields.</p> <pre> <Button Command="{Binding PowerOnInstanceCommand}" Grid.Column="0" Content="POWER ON" Horizontal <Button Command="{Binding PowerOffInstanceCommand}" Grid.Column="0" Content="POWER OFF" Horizont <Button Command="{Binding RunInstanceCommand}" Grid.Column="0" Content="RUN" HorizontalAlignme <Button Command="{Binding StopInstanceCommand}" Grid.Column="0" Content="STOP" HorizontalAlignme <Button Command="{Binding CosimulationStartCommand}" Grid.Column="2" Content="START" HorizontalA <Button Command="{Binding CosimulationStopCommand}" Grid.Column="2" Content="STOP" HorizontalAli <Button Command="{Binding CosimulationErrorCommand}" Grid.Column="2" Content="SIMULATE ERROR" Ho <Button Command="{Binding CosimulationPackageOKCommand}" Grid.Column="2" Content="PACKAGE OK" Ho <TextBox Text="{Binding StatusPLCInstance}" Name="tbPLCInstance" Grid.Column="0" VerticalAlignme <TextBox Text="{Binding StatusCoSimulation}" Name="tbCoSimulation" Grid.Column="2" VerticalAlign </Grid> <ListBox Grid.Row="4" ItemsSource="{Binding StatusListView}" ScrollViewer.HorizontalScrollBarVisibil <ListBox.ItemTemplate> </pre>

2.2.3 Programming the PLC instance

Creating an instance

Table 2-4: Class “MainWindowViewModel.cs”

No.	Description
1.	Open the class “MainWindowViewModel.cs” in the “ViewModels” folder.
2.	<p>In the #region block “Fields” a “virtualController” tag of the “PLCInstance” object type is defined.</p> <pre> /// <summary> /// PLCSIM Adv. Instance of the virtual controller /// </summary> public PLCInstance virtualController = null; </pre>
3.	<p>In the constructor of class (#region block “C’Tor”) a new object of class “PLCInstance” is created and assigned to the “virtualController” tag. “TransportControl” is transferred as name for the new instance.</p> <pre> public MainWindowViewModel() { StatusListView = new ObservableCollection<String>(); try { // New PLC instance Name: "TransportControl" virtualController = new PLCInstance("TransportControl"); } } </pre>

Table 2-5: Class “PLCInstance.cs”

No.	Description
1.	Open the “PLCInstance.cs” class in the “Models” folder.
2.	<p>In the #region block “Properties” an “instance” tag is defined as “IInstance” user interface.</p> <pre> /// <summary> /// instance of PLCSIM Adv. virtual Controller /// </summary> public IInstance instance { get; set; } </pre>
3.	<p>In the constructor of the class (#region block “C’Tor”) a new instance of a virtual controller is registered in the Runtime Manager with the function “RegisterInstance()” of the API ISimulationRuntimeManager. The function creates and returns an interface from this instance. The created interface is assigned to the “instance” tag.</p> <pre> public PLCInstance(string instanceName) { instance = SimulationRuntimeManager.RegisterInstance(instanceName); instance.IsAlwaysSendOnEndOfCycleEnabled = true; instance.CommunicationInterface = ECommunicationInterface.Softbus; instance.OnConfigurationChanged += instance_OnConfigurationChanged; instance.OnEndOfCycle += instance_OnEndOfCycle; } </pre>
4.	<p>The following settings are made in addition to the registered instance:</p> <ul style="list-style-type: none"> - “instance.CommunicationInterface = ECommunicationInterface.Softbus” sets the communication interface of the virtual controller to local communication (softbus). - “instance.IsAlwaysSendOnEndOfCycleEnabled = true” has the effect that the “OnEndOfCycle” event is triggered at the end of each CPU cycle. - The events “OnConfigurationChanged” and “OnEndOfCycle” is assigned an event handler each.

No.	Description
	<pre> public PLCInstance(string instanceName) { instance = SimulationRuntimeManager.RegisterInstance(instanceName); instance.IsAlwaysSendOnEndOfCycleEnabled = true; instance.CommunicationInterface = ECommunicationInterface.Softbus; instance.OnConfigurationChanged += instance_OnConfigurationChanged; instance.OnEndOfCycle += instance_OnEndOfCycle; } </pre>

Unregistering instance

Table 2-6: Class “MainWindow.xaml.cs”

No.	Description
1.	Open the “MainWindow.xaml.cs” class in the “Views” folder.
2.	<p>The “Window_Closing” event calls the “UnregisterInstance()” function of the API Instance. This deregisters the instance of the virtual controller again from the Runtime Manager.</p> <pre> private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e) { ViewModels.MainWindowViewModel Application = this.DataContext as ViewModels.MainWindowViewModel; //Unregister Instance Application.virtualController.instance.UnregisterInstance(); } </pre>

The application is closed with the “EXIT” button or with the “close” icon of the user interface. This triggers the “Window_Closing” event.

Switching on instance

Table 2-7: Class “MainWindowViewModel.cs”

No.	Description
1.	Open the class “MainWindowViewModel.cs” in the “ViewModels” folder.
2.	<p>By clicking on the “POWER ON” button on the user interface the method “PowerOnPLCInstance()” of the virtual controller is called in the #region block “Public Methods”.</p> <pre> /// <summary> /// Power On registred Instance of virtual controller /// </summary> public void PowerOnController() { try { WriteStatusEntry(String.Format("Power On Instance: {0}", virtualController.instance.Name)); virtualController.PowerOnPLCInstance(); } catch (SimulationRuntimeException simRtEx) { WriteStatusEntry(String.Format("PowerOn Instance failed: {0}", simRtEx.Message)); } } </pre>

Table 2-8: Class “PLCInstance.cs”

No.	Description
1.	Open the “PLCInstance.cs” class in the “Models” folder.
2.	<p>The method “PowerOnPLCInstance()” in the #region block “Public Methods” calls the functions “PowerOn()” and “SetIPSuite()” of the API Instance. The “PowerOn()” function creates the process for the simulation runtime instance and starts the firmware of the virtual controller. As timeout value 60.000 ms have been configured. The “SetIPSuite()” function configures the network interface of the virtual</p>

No.	Description
	<p>controller.</p> <pre> /// <summary> /// Power On PLCSIM Adv. Instanz, set IPSuite of instance /// </summary> /// <returns></returns> public void PowerOnPLCInstance() { instance.PowerOn(60000); instance.SetIPSuite(0, instanceIP, true); } </pre>
3.	<p>The parameters (IP address, subnet mask and standard gateway) of the network interface are defined in the #region block "Properties" in the "instanceIP" tag of the "SIPSuite4" structure type.</p> <pre> private SIPSuite4 instanceIP = new SIPSuite4("192.168.0.101", "255.255.255.0", "0.0.0.0"); </pre>

Switching off instance

Table 2-9: Class "MainWindowViewModel.cs"

No.	Description
1.	Open the class "MainWindowViewModel.cs" in the "ViewModels" folder.
2.	<p>By clicking the "POWER OFF" button on the user interface, the method "PowerOffPLCInstance()" of the virtual controller is called in the #region block "Public Methods".</p> <pre> /// <summary> /// Power Off registred Instance of virtual controller /// </summary> public void PowerOffController() { try { WriteStatusEntry(String.Format("Power Off Instance: {0}", virtualController.instance.Name)); virtualController.PowerOffPLCInstance(); } catch (SimulationRuntimeExpection simRtEx) { WriteStatusEntry(String.Format("PowerOff Instance failed: {0}", simRtEx.Message)); } } </pre>

Table 2-10: Class "PLCInstance.cs"

No.	Description
1.	Open the "PLCInstance.cs" class in the "Models" folder.
2.	<p>The method "PowerOffPLCInstance()" in the #region block "Public Methods" calls the function "PowerOff()" of the API Instance. Shuts down the simulation runtime and closes its process. As timeout value 6.000 ms have been configured.</p> <pre> /// <summary> /// Power Off PLCSIM Adv. Instance /// </summary> public void PowerOffPLCInstance() { instance.PowerOff(6000); } </pre>

Starting instance

Table 2-11: Class "MainWindowViewModel.cs"

No.	Description
1.	Open the class "MainWindowViewModel.cs" in the "ViewModels" folder.
2.	<p>By clicking on the "RUN" button on the user interface, the method "RunPLCInstance()" of the virtual controller is called in the #region block "Public Methods".</p> <pre> /// <summary> /// Run registered Instance of virtual controller /// </summary> public void RunController() { try { WriteStatusEntry(Severity.Format("Run Instance: {0}", virtualController.Instance.Name)); virtualController.RunPLCInstance(); } catch (SimulationRuntimeExcpetion simRtEx) { WriteStatusEntry(Severity.Format("Run Instance failed: {0}! Please load plc program before execute RUN.", simRtEx.Message)); } } </pre>

Table 2-12: Class "PLCInstance.cs"

No.	Description
1.	Open the "PLCInstance.cs" class in the "Models" folder.
2.	<p>The method "RunPLCInstance()" in the #region block "Public Methods" calls the function "Run()" of the API Instance. This function requests the virtual controller to go to the RUN mode. As timeout value 6.000 ms have been configured.</p> <pre> /// <summary> /// Run PLCSIM Adv. Instance /// </summary> public void RunPLCInstance() { instance.Run(6000); } </pre>

Stopping instance

Table 2-13: Class "MainWindowViewModel.cs"

No.	Description
1.	Open the class "MainWindowViewModel.cs" in the "ViewModels" folder.
2.	<p>By clicking on the "STOP" button on the user interface, the method "StopPLCInstance()" of the virtual controller is called in the #region block "Public Methods".</p> <pre> /// <summary> /// Stop registred Instance of virtual controller /// </summary> public void StopController() { try { WriteStatusEntry(String.Format("Stop Instance: {0}", virtualController.instance.Name)); virtualController.StopPLCInstance(); } catch (SimulationRuntimeExpection simRtEx) { WriteStatusEntry(String.Format("Stop Instance failed: {0}", simRtEx.Message)); } } </pre>

Table 2-14: Class "PLCInstance.cs"

No.	Description
1.	Open the "PLCInstance.cs" class in the "Models" folder.
2.	<p>The method "StopPLCInstance()" in the #region block "Public Methods" calls the function "Stop()" of the API Instance. This function requests the virtual controller to go to the STOP mode. As timeout value 6.000 ms have been configured.</p> <pre> /// <summary> /// Stop PLCSIM Adv. Instance /// </summary> public void StopPLCInstance() { instance.Stop(6000); } #endregion //Public Methods </pre>

Exchanging I/O data

Table 2-15: Class "PLCInstance.cs"

No.	Description
1.	Open the "PLCInstance.cs" class in the "Models" folder.
2.	<p>The event handler "instance_OnConfigurationChanged" in the #region block "Events" calls the function "UpdateTagList()" of the API Instance. This function reads the tags from the virtual controller and writes them, ordered by name, into the joint memory.</p> <p>In order to read the input and output tags of the virtual controller, the "ETagListDetails.IO" parameter is transferred to the function.</p> <pre> /// <summary> /// Event when Configuration changed of the PLC (during download) /// </summary> /// <param name="in_Sender"> PLC which fired this event</param> /// <param name="in_ErrorCode"> ErrorCode of Runtime of the PLC</param> /// <param name="in_DateTime"> DateTime when the configuration changed</param> void instance_OnConfigurationChanged(IInstance in_Sender, ERuntimeErrorCode in_ErrorCode, DateTime in_DateTime, InstanceConfigChanged in_InstanceConfigChanged, uint in_Param1, uint in_Param2, uint in_Param3, uint in_Param4) { IsConfigured = false; try { instance.UpdateTagList(ETagListDetails.IO); IsConfigured = true; } catch (Exception ex) { } } </pre>
3.	<p>The event handler "instance_OnEndOfCycle" in the #region block "Events" calls the functions "ReadBool()" and "WriteBool()" of the API Instance. Thus Boolean tags of the virtual controller are read or written via the tag name.</p> <pre> void instance_OnEndOfCycle(IInstance in_Sender, ERuntimeErrorCode in_ErrorCode, DateTime in_DateTime, long in_CycleTime_ns, uint in_CycleCount) { if (IsConfigured) { try { // Read outputs of the virtual controller and assign to variables of the Co-Simulation coSimulation.setOnBeltActive = instance.ReadBool("setOnBeltActive"); coSimulation.moveBeltActive = instance.ReadBool("moveBeltActive"); coSimulation.setOffBeltActive = instance.ReadBool("setOffBeltActive"); coSimulation.releaseActive = instance.ReadBool("releaseActive"); coSimulation.acknowledgeActive = instance.ReadBool("acknowledgeActive"); coSimulation.restartActive = instance.ReadBool("restartActive"); // Call the Co-Simulation programm coSimulation.CoSimProgramm(); // Write the Co-Simulation values to the inputs of the virtual controller instance.WriteBool("sensorStartPos", coSimulation.sensorStartPos); instance.WriteBool("sensorBeltStart", coSimulation.sensorBeltStart); instance.WriteBool("sensorBeltDest", coSimulation.sensorBeltDest); instance.WriteBool("sensorEndPos", coSimulation.sensorEndPos); } catch (Exception ex) { } } } </pre>

2.2.4 Programming the co-simulation

Simulation parameters

Table 2-16: Class "MainWindowViewModel.cs"

No.	Description
1.	Open the class "MainWindowViewModel.cs" in the "ViewModels" folder.
2.	In the #region block "Fields" a "transportSystem" tag of the "Cosimulation" object type is defined. <pre>public Cosimulation transportSystem = null;</pre>
3.	In the constructor of class (#region block "C'tor") a new object "transportSystem" of the class "Cosimulation" is created. Simulation parameters: <ul style="list-style-type: none"> - Simulation time of the robot arms and drives: 2.000 ms - Simulation time of the sensors: 1.000ms <pre>// New Cosimulation "transportSystem" // Simulation time movement 2000ms // Simulation time sensor 1000ms transportSystem = new Cosimulation(2000, 1000);</pre>

Data exchange

Table 2-17: Class "MainWindowViewModel.cs"

No.	Description
1.	Open the class "MainWindowViewModel.cs" in the "ViewModels" folder.
2.	In the constructor of class (#region block "C'tor") a reference to the "coSimulation" object of the virtual controller is stored to the "transportSystem" object. <pre>virtualController.coSimulation = transportSystem;</pre>

Table 2-18: Class "PLCInstance.cs"

No.	Description
1.	Open the "PLCInstance.cs" class in the "Models" folder.
2.	In the event handler "instance_OnEndOfCycle" in the #region block "Events" the read output tags of the virtual controller are transferred to the co-simulation (1). Subsequently, there is a call of the co-simulation program "CoSimProgramm()" (2). Finally, the simulated values of the co-simulation are written to the input tags of the virtual controller (3). <pre>// Read outputs of the PLC and assign to variables of the Co-Simulation coSimulation.setOnBeltActive = instance.ReadBool("setOnBeltActive"); coSimulation.moveBeltActive = instance.ReadBool("moveBeltActive"); coSimulation.setOffBeltActive = instance.ReadBool("setOffBeltActive"); coSimulation.releaseActive = instance.ReadBool("releaseActive"); coSimulation.acknowledgeActive = instance.ReadBool("acknowledgeActive"); coSimulation.restartActive = instance.ReadBool("restartActive"); // Call the Co-Simulation programm coSimulation.CoSimProgramm(); // Write the Co-Simulation values to the inputs of the virtual controller instance.WriteBool("sensorStartPos", coSimulation.sensorStartPos); instance.WriteBool("sensorBeltStart", coSimulation.sensorBeltStart); instance.WriteBool("sensorBeltDest", coSimulation.sensorBeltDest); instance.WriteBool("sensorEndPos", coSimulation.sensorEndPos);</pre>

Simulation program

Table 2-19: Class “Cosimulation.cs”

No.	Description
1.	Open the “Cosimulation.cs” class in the “Models” folder.
2.	<p>In the declaration of the class, Boolean tags are defined for the data exchange with the virtual controller.</p> <pre> //--// Inputs (Outputs of the virtual controller) public bool setOnBeltActive; public bool moveBeltActive; public bool setOffBeltActive; public bool releaseActive; public bool acknowledgeActive; public bool restartActive; //--// //--// Outputs (Inputs of the virtual controller) public bool sensorStartPos = false; public bool sensorBeltStart = false; public bool sensorBeltDest = false; public bool sensorEndPos = false; //--// </pre>
3.	<p>In the method “CoSimProgramm()” in the #region block “Cosimulation” the states of the simulated system are programmed in a switch-case instruction. A more detailed description can be found in chapter 3.2.2 or in the code comments.</p> <pre> public void CoSimProgramm() { if (restartActive) // Restart command from the virtual controller { step = 0; // Set start step startedTimer = false; // Reset indicator for started timer OnErrorSimulationStateChanged(this, new PropertyChangedEventArgs("Black")); // Reset "SIMULATE ERROR" button color } if (run) // Active when "START" button pressed for Co-Simulation { OnOperatingStateChanged(this, new PropertyChangedEventArgs("ACTIVE")); // Shows "ACTIVE" state of the Co-Simulation // Reset all sensors at every call (sensors are set in the corresponding case) sensorStartPos = false; sensorBeltStart = false; sensorBeltDest = false; sensorEndPos = false; switch (step) { case 0: // Simulation of sensor: Package on start position sensorStartPos = true; // Sensor active if (!startedTimer) // Starts once the simulation time for the sensor { sensorTimer.Start(); // Start timer for sensor simulation startedTimer = true; // Set indicator for started timer } if (setOnBeltActive & nextStep) // Next step after sensor simulation time elapsed and command from virtual controller { step = 1; // Set number of the next step nextStep = false; // Reset next step variable startedTimer = false; // Reset indicator for started timer } break; </pre>

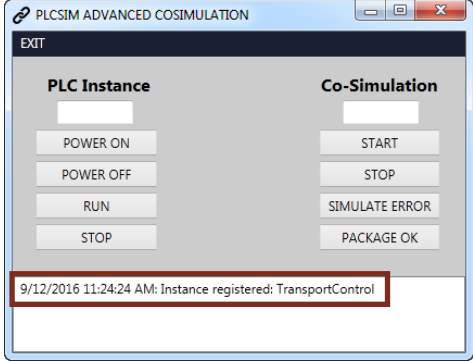
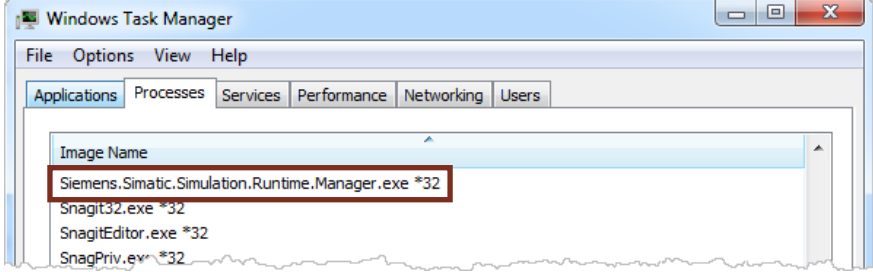
2.3 Operating the Application Example

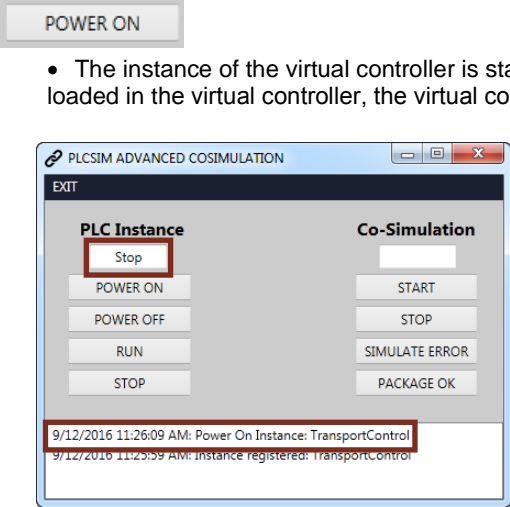
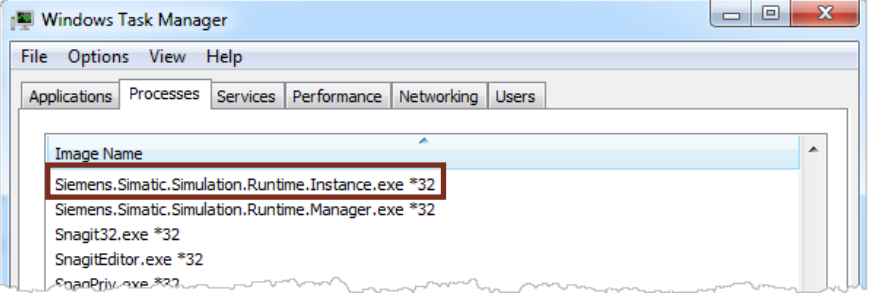
Note

In order to start the PLCSIM Advanced simulation via the “CoSimulationPlcSimAdv” application, you do not have to start the control panel of PLCSIM Advanced.

2.3.1 Switching on virtual controller

Table 2-20: Switching on virtual controller

No.	Action
1.	<p>Unzip the file “109739660_PLCSIM_Advanced_COSIM_APPL_V10.zip”. Open “CoSimulationPlcSimAdv > bin > Release”. Start the application “CoSimulationPlcSimAdv.exe”.</p> <p>The user interface of the application opens. You can see the information that the instance “TransportControl” has been registered in the status list.</p> 
2.	<p>Optional: Start the Windows Task Manager and check the started process of the Runtime Manager in “Processes”.</p> 

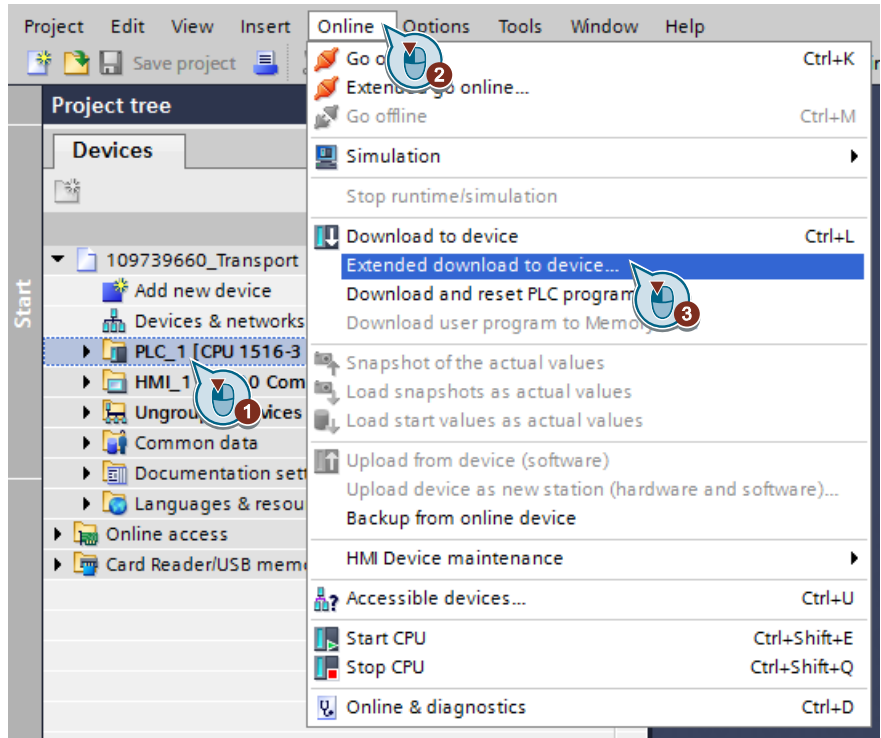
No.	Action
3.	<p>Click on the "POWER ON" button in the user interface to start the virtual controller.</p>  <ul style="list-style-type: none"> The instance of the virtual controller is started. If no program has been loaded in the virtual controller, the virtual controller is in "Stop" mode. <p>Note If you have started the application before and have already loaded a program in the virtual controller, an automatic upload from the virtual SIMATIC memory card takes place and the virtual controller is in "Run" mode.</p>
4.	<p>Optional: Start the Windows Task Manager and check the started process of the Runtime Instance in "Processes".</p> 

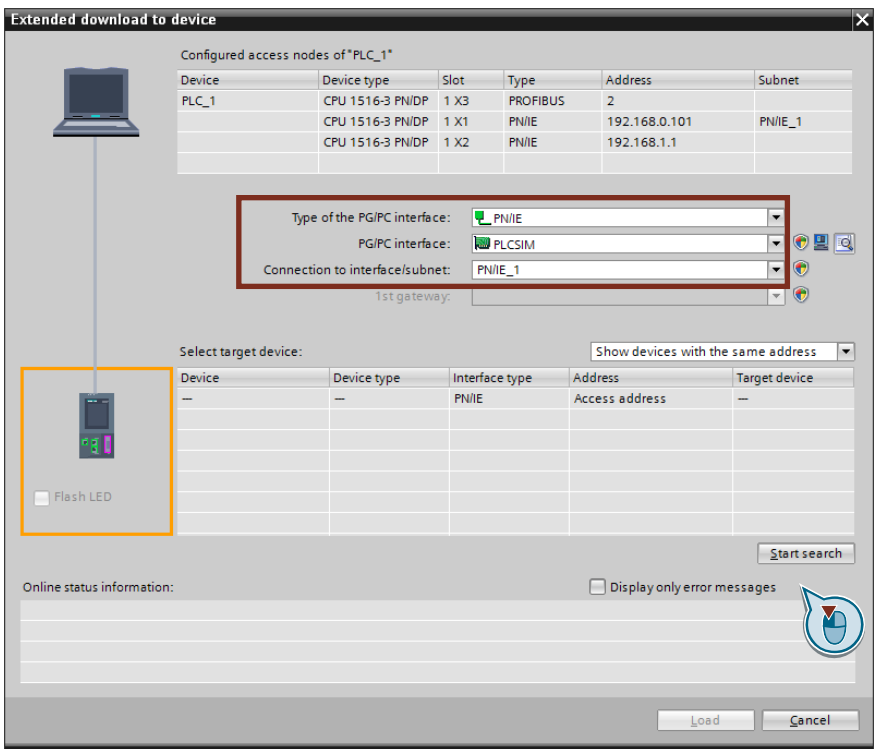
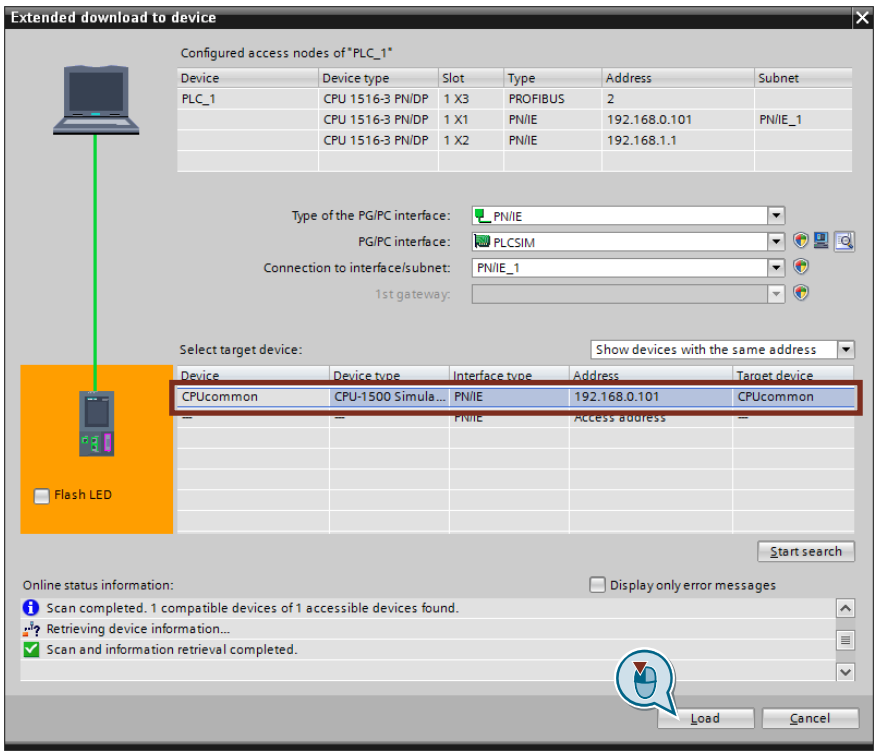
2.3.2 Loading TIA Portal project into the virtual controller

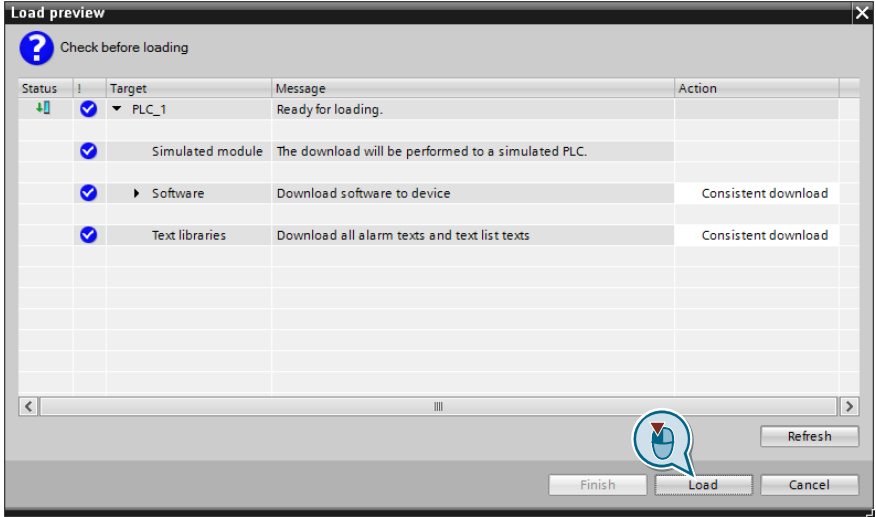
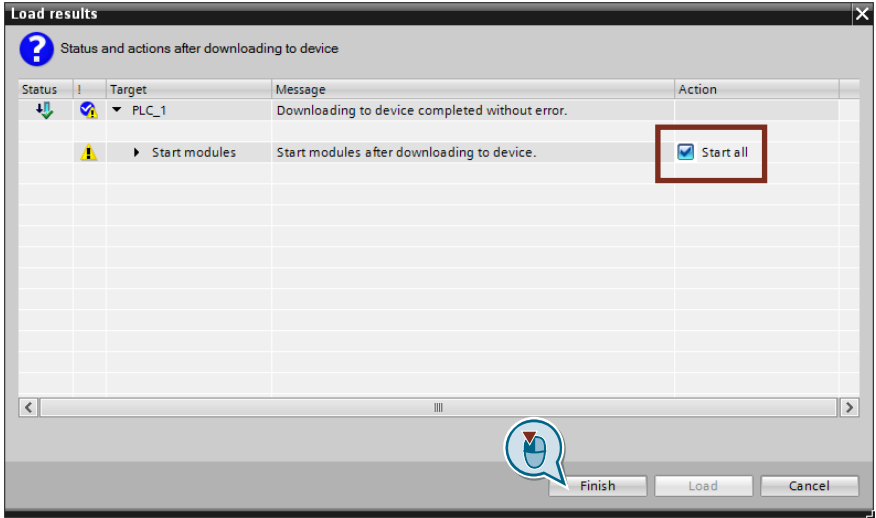
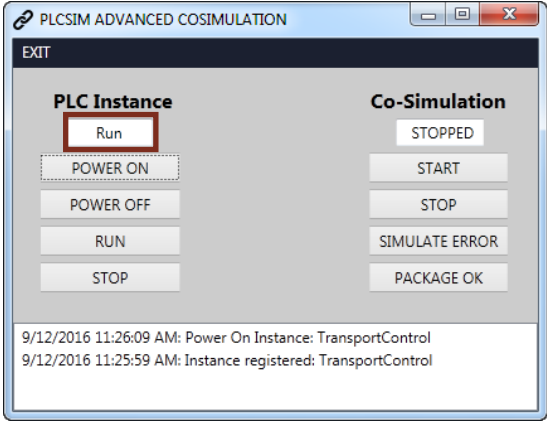
Note

In order to load the TIA Portal project into the virtual controller, first you have to switch on the virtual controller (see chapter 2.3.1).

Table 2-21: Loading TIA Portal project into the virtual controller

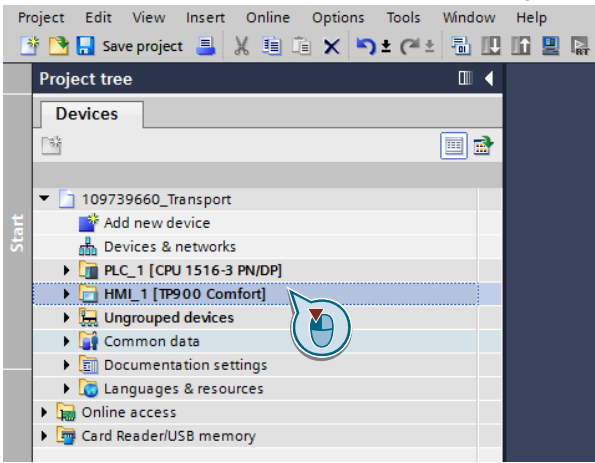

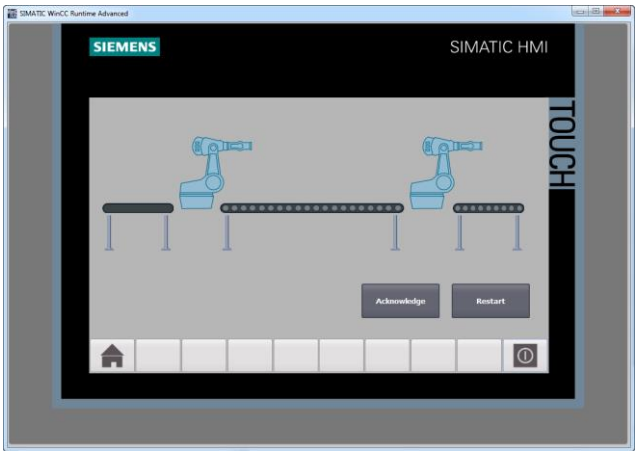
No.	Action
1.	Open the TIA Portal project “109739660_Transport” from the download included “109739660_PLCSIM_Advanced_S7_PROJ_V10.zip”.
2.	<ol style="list-style-type: none"> 1. Select the PLC_1 from the project navigation. 2. Open the “Online” menu. 3. Select “Extended download to device...”. 

No.	Action																																							
3.	<p>Set "PLCSIM" as "PG/PC interface" and click on "Start search".</p>  <p>Extended download to device</p> <p>Configured access nodes of "PLC_1"</p> <table border="1"> <thead> <tr> <th>Device</th> <th>Device type</th> <th>Slot</th> <th>Type</th> <th>Address</th> <th>Subnet</th> </tr> </thead> <tbody> <tr> <td>PLC_1</td> <td>CPU 1516-3 PN/DP</td> <td>1 X3</td> <td>PROFIBUS</td> <td>2</td> <td></td> </tr> <tr> <td></td> <td>CPU 1516-3 PN/DP</td> <td>1 X1</td> <td>PN/IE</td> <td>192.168.0.101</td> <td>PN/IE_1</td> </tr> <tr> <td></td> <td>CPU 1516-3 PN/DP</td> <td>1 X2</td> <td>PN/IE</td> <td>192.168.1.1</td> <td></td> </tr> </tbody> </table> <p>Type of the PG/PC interface: <input type="text" value="PN/IE"/></p> <p>PG/PC interface: <input type="text" value="PLCSIM"/></p> <p>Connection to interface/subnet: <input type="text" value="PN/IE_1"/></p> <p>1st gateway: <input type="text"/></p> <p>Select target device: <input type="text" value="Show devices with the same address"/></p> <table border="1"> <thead> <tr> <th>Device</th> <th>Device type</th> <th>Interface type</th> <th>Address</th> <th>Target device</th> </tr> </thead> <tbody> <tr> <td>—</td> <td>—</td> <td>PN/IE</td> <td>Access address</td> <td>—</td> </tr> </tbody> </table> <p><input type="checkbox"/> Flash LED</p> <p><input type="button" value="Start search"/></p> <p>Online status information: <input type="checkbox"/> Display only error messages</p> <p><input type="button" value="Load"/> <input type="button" value="Cancel"/></p>	Device	Device type	Slot	Type	Address	Subnet	PLC_1	CPU 1516-3 PN/DP	1 X3	PROFIBUS	2			CPU 1516-3 PN/DP	1 X1	PN/IE	192.168.0.101	PN/IE_1		CPU 1516-3 PN/DP	1 X2	PN/IE	192.168.1.1		Device	Device type	Interface type	Address	Target device	—	—	PN/IE	Access address	—					
Device	Device type	Slot	Type	Address	Subnet																																			
PLC_1	CPU 1516-3 PN/DP	1 X3	PROFIBUS	2																																				
	CPU 1516-3 PN/DP	1 X1	PN/IE	192.168.0.101	PN/IE_1																																			
	CPU 1516-3 PN/DP	1 X2	PN/IE	192.168.1.1																																				
Device	Device type	Interface type	Address	Target device																																				
—	—	PN/IE	Access address	—																																				
4.	<p>Select the simulated CPU and click "Load".</p>  <p>Extended download to device</p> <p>Configured access nodes of "PLC_1"</p> <table border="1"> <thead> <tr> <th>Device</th> <th>Device type</th> <th>Slot</th> <th>Type</th> <th>Address</th> <th>Subnet</th> </tr> </thead> <tbody> <tr> <td>PLC_1</td> <td>CPU 1516-3 PN/DP</td> <td>1 X3</td> <td>PROFIBUS</td> <td>2</td> <td></td> </tr> <tr> <td></td> <td>CPU 1516-3 PN/DP</td> <td>1 X1</td> <td>PN/IE</td> <td>192.168.0.101</td> <td>PN/IE_1</td> </tr> <tr> <td></td> <td>CPU 1516-3 PN/DP</td> <td>1 X2</td> <td>PN/IE</td> <td>192.168.1.1</td> <td></td> </tr> </tbody> </table> <p>Type of the PG/PC interface: <input type="text" value="PN/IE"/></p> <p>PG/PC interface: <input type="text" value="PLCSIM"/></p> <p>Connection to interface/subnet: <input type="text" value="PN/IE_1"/></p> <p>1st gateway: <input type="text"/></p> <p>Select target device: <input type="text" value="Show devices with the same address"/></p> <table border="1"> <thead> <tr> <th>Device</th> <th>Device type</th> <th>Interface type</th> <th>Address</th> <th>Target device</th> </tr> </thead> <tbody> <tr> <td>CPUcommon</td> <td>CPU-1500 Simula...</td> <td>PN/IE</td> <td>192.168.0.101</td> <td>CPUcommon</td> </tr> <tr> <td>—</td> <td>—</td> <td>PN/IE</td> <td>Access address</td> <td>—</td> </tr> </tbody> </table> <p><input checked="" type="checkbox"/> Flash LED</p> <p><input type="button" value="Start search"/></p> <p>Online status information: <input type="checkbox"/> Display only error messages</p> <p>Scan completed. 1 compatible devices of 1 accessible devices found.</p> <p>Retrieving device information...</p> <p>Scan and information retrieval completed.</p> <p><input type="button" value="Load"/> <input type="button" value="Cancel"/></p>	Device	Device type	Slot	Type	Address	Subnet	PLC_1	CPU 1516-3 PN/DP	1 X3	PROFIBUS	2			CPU 1516-3 PN/DP	1 X1	PN/IE	192.168.0.101	PN/IE_1		CPU 1516-3 PN/DP	1 X2	PN/IE	192.168.1.1		Device	Device type	Interface type	Address	Target device	CPUcommon	CPU-1500 Simula...	PN/IE	192.168.0.101	CPUcommon	—	—	PN/IE	Access address	—
Device	Device type	Slot	Type	Address	Subnet																																			
PLC_1	CPU 1516-3 PN/DP	1 X3	PROFIBUS	2																																				
	CPU 1516-3 PN/DP	1 X1	PN/IE	192.168.0.101	PN/IE_1																																			
	CPU 1516-3 PN/DP	1 X2	PN/IE	192.168.1.1																																				
Device	Device type	Interface type	Address	Target device																																				
CPUcommon	CPU-1500 Simula...	PN/IE	192.168.0.101	CPUcommon																																				
—	—	PN/IE	Access address	—																																				

No.	Action
5.	<p>For the message that follows, click on “Load”.</p> 
6.	<p>Enable the “Start all” option box and click on “Finish”.</p> 
7.	<p>After the download, check whether the virtual controller has changed to the “Run” mode.</p> 

2.3.3 Starting WinCC Runtime

Table 2-22: Starting WinCC Runtime

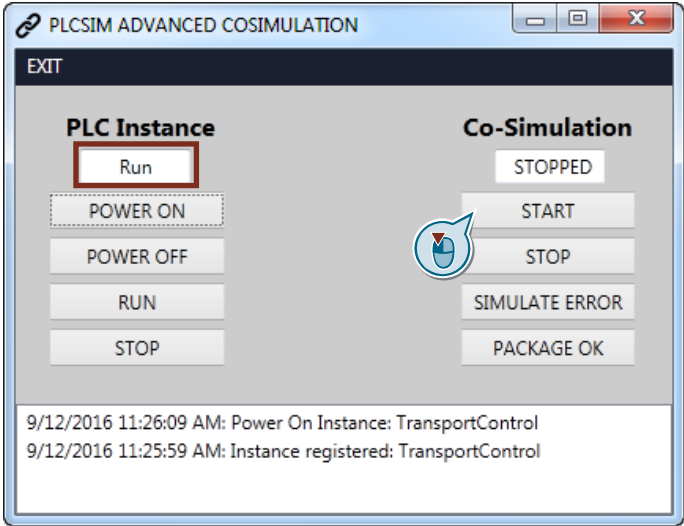
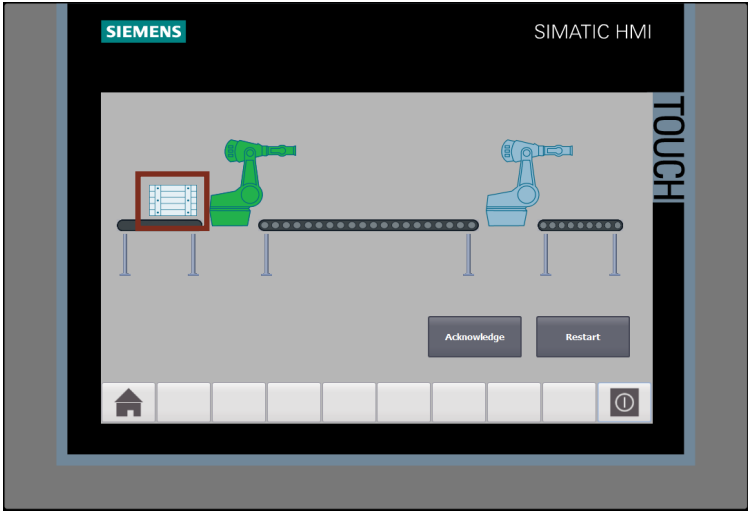
No.	Action
1.	<p>Select the HMI_1 in TIA Portal in the project navigation</p> 
2.	<p>Click on “Start simulation” in the toolbar of the TIA Portal.</p>  <ul style="list-style-type: none">• After the compilation, the WinCC Runtime of the SIMATIC HMI starts. 

2.3.4 Starting the co-simulation

Note

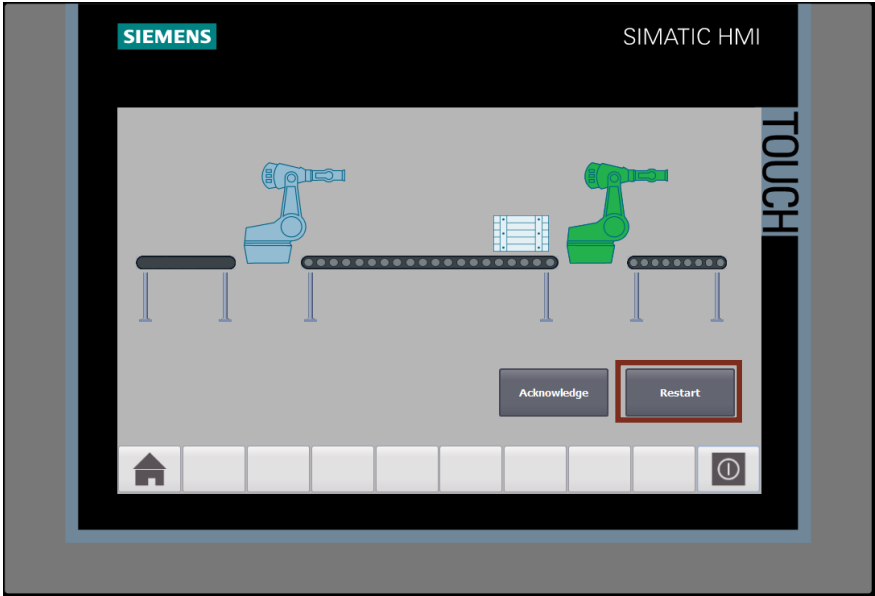
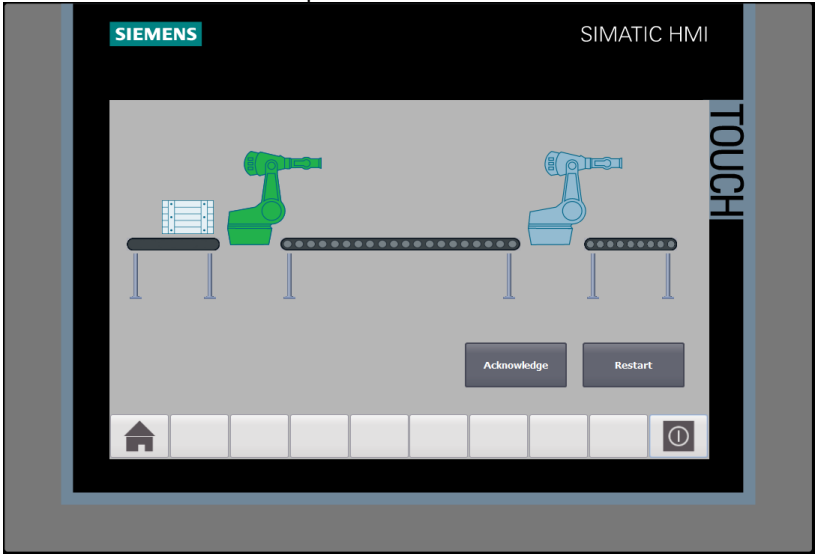
Set the virtual controller to “Run” mode to operate the co-simulation.

Table 2-23: Starting the co-simulation

No.	Action
1.	<div>Click “START” in the user interface of the application to start the co-simulation.</div> <div></div> <div><ul style="list-style-type: none">• The co-simulation places a package at the starting position and the conveyor system starts moving. Once a package has been transported away, a new crate is placed at the starting position.</div> <div></div>
2.	Verify the function of the conveyor system in the simulated HMI screen.

2.3.5 Restarting the conveyor system

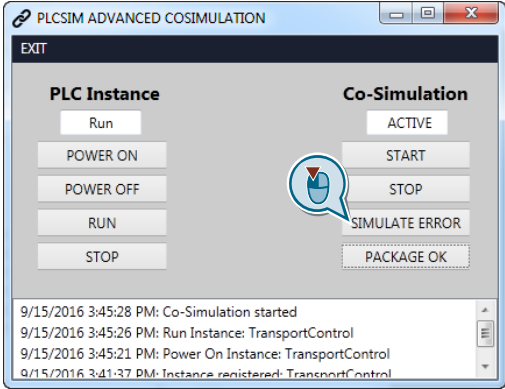

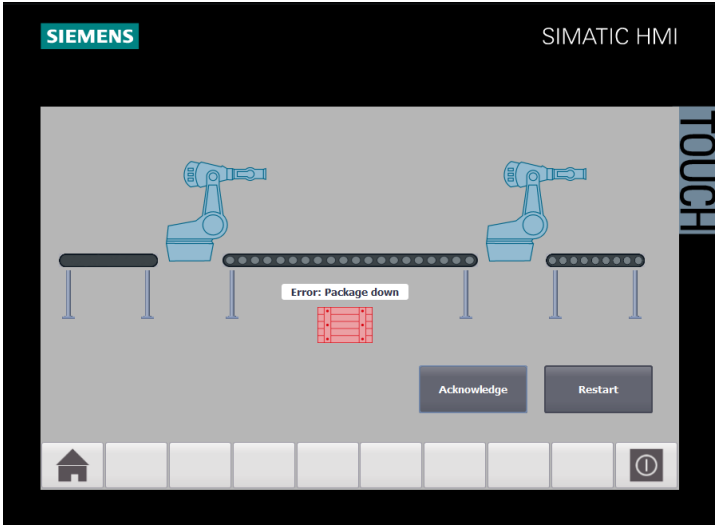
Table 2-24: Restarting system

No.	Action
1.	<div>Click the “Restart” button in the simulated HMI screen.</div> <div></div> <div><ul style="list-style-type: none">The co-simulation places the package back at the starting position and the controller is initialized. Transportation is restarted.</div> <div></div>

2.3.6 Simulating an error

Note Set the virtual controller to “Run” mode to operate the co-simulation.

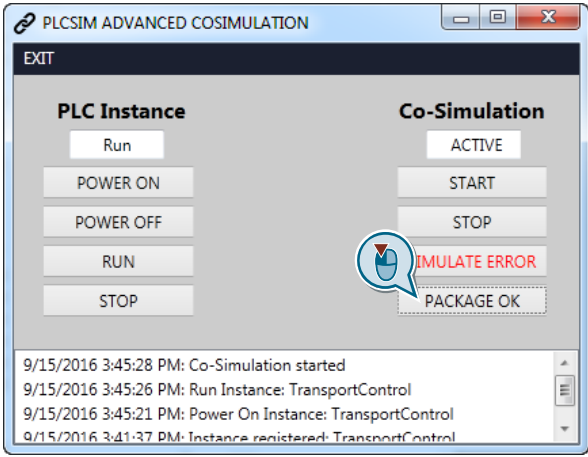
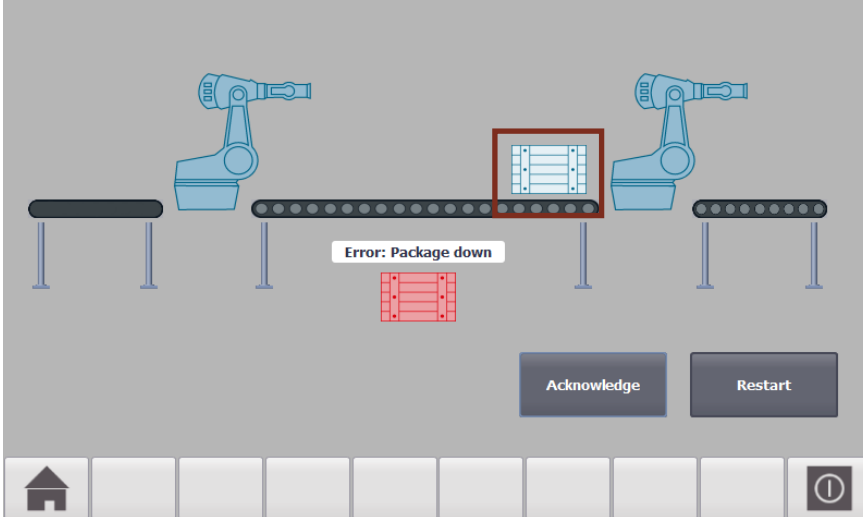
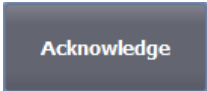
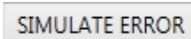
Table 2-25: Simulating an error

No.	Action
1.	<p>To simulate the falling off of a package from transport belt 1, click on the “SIMULATE ERROR” button in the user interface of the application during an active co-simulation.</p> <div></div> <ul style="list-style-type: none">• An activated error simulation is displayed by a red font color. <div></div>
2.	<p>Verify the following states in the simulated HMI screen:</p> <ul style="list-style-type: none">• The package does not reach its target position before the second robot arm within the configured monitoring time• The “Error: Package down” is show. <div></div>
3.	<p>You have the following options to remove the error:</p> <ul style="list-style-type: none">• Restarting the (chapter 2.3.5)• Removing and acknowledging error (see chapter 2.3.7)

2.3.7 Removing and acknowledging error

Note Set the virtual controller to “Run” mode to operate the co-simulation.

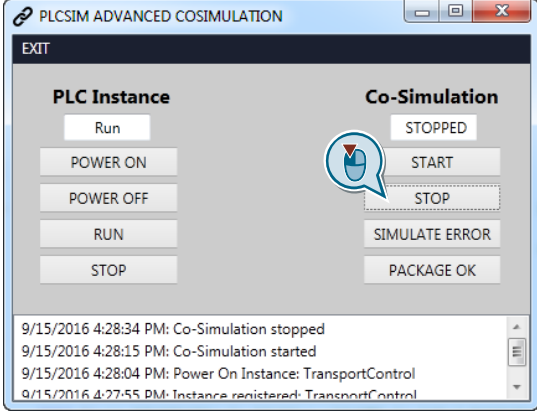
Table 2-26: Removing and acknowledging error

No.	Action
1.	<p>In the user interface of the application, click on “PACKAGE OK”. This simulates manual picking up and placing the package on transport belt 1.</p> 
2.	<p>Verify that the package has been placed in front of the second robot arm.</p> 
3.	<p>Click the “Acknowledge” button in the simulated HMI screen to acknowledge the error and to continue the transport.</p> 
4.	<p>Check whether the transport is continued and the font of the “SIMULATE ERROR” in the user interface appears in black again.</p> 

2.3.8 Stopping the co-simulation

Note Set the virtual controller to “Run” mode to operate the co-simulation.

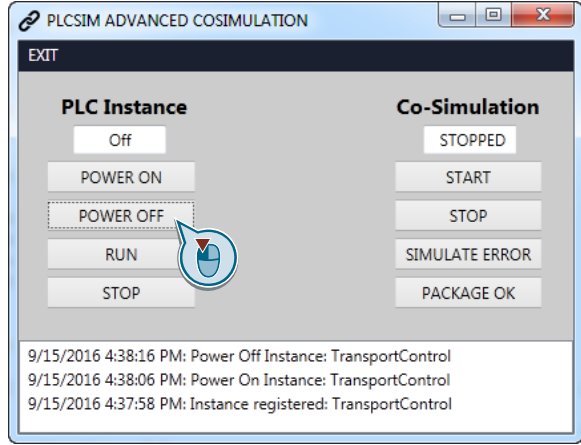
Table 2-27: Stopping the co-simulation

No.	Action
1.	<div>In the user interface of the application, click on the “STOP” button.<div></div><ul style="list-style-type: none">The co-simulation is stopped in the current step.</div>

Note When you stop the co-simulation while the package is on transport belt 1, the monitoring time for transport belt 1 can be exceeded in the controller. As a result, the error of a fallen down package is displayed. In this case, restart the conveyor system (chapter 2.3.5), to resynchronize the co-simulation and the system control.

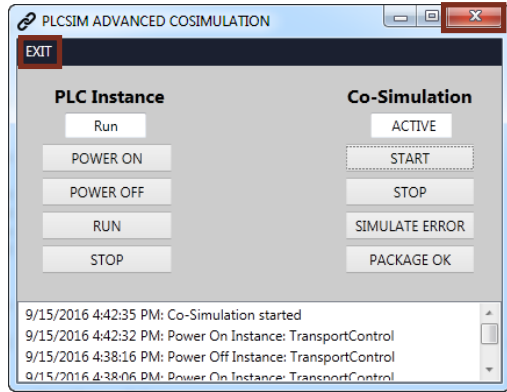
2.3.9 Switching off virtual controller

Table 2-28: Switching off virtual controller

No.	Action
1.	<p>Click the "POWER OFF" button in the user interface of the application.</p> 
2.	<p>Optional: Check whether the virtual controller is switched off and the process of the Runtime Instance in the Windows Task Manager has been ended.</p>

2.3.10 Closing application

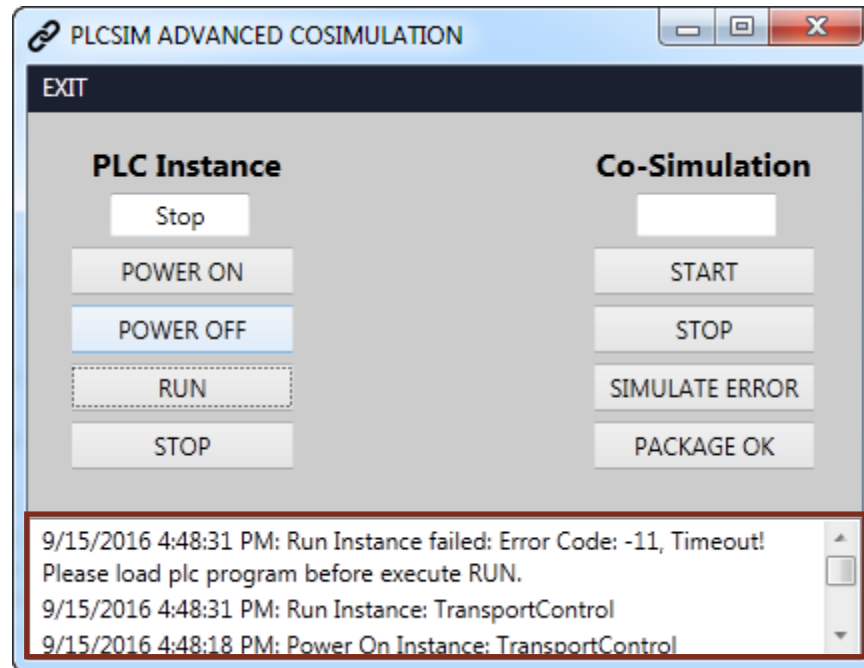
Table 2-29: Closing application

No.	Action
1.	<p>In the user interface of the application, click on the "EXIT" button or the "close" icon to close and exit the application.</p> 
2.	<p>Optional: Check whether the process of the Runtime Manager and the Runtime Instance in the Windows Task Manager has been ended.</p>

2.3.11 Error handling

In the list box of the user interface the errors are shown with a note indicating the cause.

Figure 2-2: List box



3 Valuable Information

3.1 Basics of S7-PLCSIM Advanced

This chapter includes basics of S7-PLCSIM Advanced. Detailed information can be found in the function manual "SIMATIC S7-PLCSIM Advanced" \3\.

<https://support.industry.siemens.com/cs/ww/en/view/109739153>

3.1.1 Installation

If you intend to execute several instances of PLCSIM Advanced in parallel or simulate the communication between PLCSIM Advanced V1.0 and operator panels as of version 14.0, you require powerful computer hardware.

Requirements for the installation

Before you start the installation, the following requirements have to be fulfilled:

- Hardware and software of the PC or of the SIMATIC Field PG fulfill the system requirements (see function manual "SIMATIC S7-PLCSIM Advanced", chapter "[System requirements](#)").
- The person who performs the installation has the administrator rights on the respective computer.

3.1 Basics of S7-PLCSIM Advanced

- No other programs are active. This also applies to the Siemens Automation License Manager and other Siemens applications.
- All S7 PLCSIM versions as of V12 have been uninstalled.

Note

A requirement for the installation of SIMATIC S7-PLCSIM Advanced is that no other S7-PLCSIM installation is on the same computer.

3.1.2 Difference between S7-PLCSIM V14 and PLCSIM Advanced V1.0

Table 3-1: Difference between PLCSIM Advanced and PLCSIM V14

Function	PLCSIM Advanced V1.0	PLCSIM V14
Runtime	Independent	Together with STEP 7
User interface	Control Panel	Look&Feel of TIA Portal
Communication	Softbus, TCP/IP	Only softbus
Supported CPU families	S7-1500(C,T,F), ET 200SP and ET 200SPF	S7-1200(F), S7-1500(C,T,F), ET 200SP and ET 200SPF
API for co-simulation	✓	✗
Web server	✓	✗
OPC UA	✓	✗
Process diagnostics	✓	✓
S7 communication	✓	Via softbus
Open user communication	✓	Via softbus
Traces ¹	✓	(✓)
Motion ²	✓	(✓)
Protected blocks (KHP)	✓	✗
Multiple instances	Up to 16	Up to 2
Distributed instances	✓	✗
Virtual time	✓	✗
Connection of real CPUs/HMIs	✓	✗
DNS use	✓	✗
Virtual memory card	✓	✗


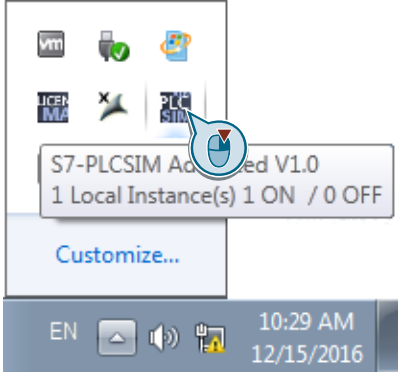
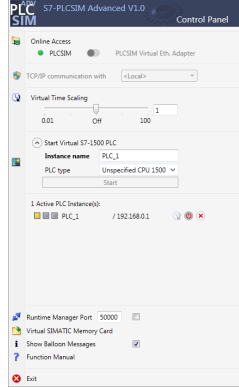
¹ For PLCSIM V14 this can be monitored in the TIA Portal; for PLCSIM Advanced V1.0 it can additionally be monitored in the web server.

² For PLCSIM V14 the axes are always in simulation mode; for PLCSIM Advanced V1.0 the axes can also be operated in "Real" mode.

3.1.3 User interface

Starting PLCSIM Advanced

Table 3-2: Starting PLCSIM Advanced

No.	Action
1.	<p>Double click the PLCSIM Advanced icon on the Windows desktop to start the program.</p> 
2.	<p>Right-click the PLCSIM Advanced icon in the info area of the task bar.</p> <p>Note: You can permanently show the icon in the info area of the task bar via the Windows functions.</p>  <p>The graphic interface of PLCSIM Advanced (Control Panel) is opened. Clicking on a free space on the desktop closes the graphic interface again.</p> 

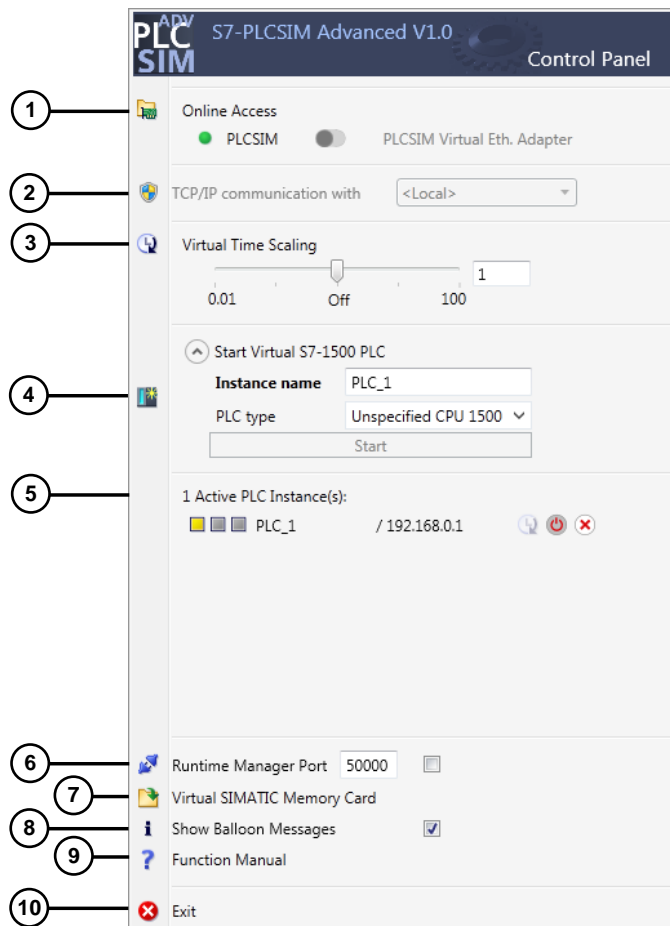
Control Panel

The control panel is optional and is not required for the operation of S7-PLCSIM Advanced via the API.

3 Valuable Information

3.1 Basics of S7-PLCSIM Advanced

Figure 3-1: Control Panel



The following table shows the basic functions of the control panel from [Figure 3-1](#):

Table 3-3: Functions of the control panel

No.	Function
1.	Switch for selecting the communication interface. <ul style="list-style-type: none"> PLCSIM corresponds to the local communication via softbus. PLCSIM Virtual Ethernet Adapter corresponds to the communication via TCP/IP.
2.	Selection of the network card for the distributed communication.
3.	Select the scaling factor for the virtual time with the slider.
4.	Enter a unique name (at least three characters) for the instance under the "Instance name" . If there is already a name in the directory of the virtual SIMATIC memory card, the already existing instance is started. Selected the CPU type to be simulated under "PLC Type" . In addition, the input fields IP address , subnet mask and default gateway become visible here when you switch the communication interface to "PLCSIM Virtual Ethernet Adapter" .
5.	The list shows the locally existing instances. The meaning of the LED and the buttons are shown when you move the mouse pointer over them.
6.	Here, you open a port to the local PC. Via the set port a remote connection to another Runtime Manager can be established.
7.	Open an Explorer window here with the path to the virtual memory card.

No.	Function
8.	Here, disable the PLCSIM Advanced messages in the Windows taskbar for the duration of the operation.
9.	Open the function manual "SIMATIC S7-PLCSIM Advanced" in the standard PDF viewer here.
10.	Switches off all local instances and deregisters you from the Runtime Manager.

3.1.4 General properties

Fields of application

- Function test of the STEP 7 program – also in the context of a system/machine
- Virtual function validation right up to virtual commissioning
- Training for S7-1500 automation without hardware (operator training)
- Factory Acceptance Test (FAT)

Advantages

The use of S7-PLCSIM Advanced offers several advantages:

- Early error detection and validation of functionality → High quality of the STEP 7 program code
- Nor real S7-CPU hardware necessary → Saving of hardware costs
- Efficiency enhancement through optimization of program parts
- Shortened development times or of time-to-market
- Reducing risks for commissioning
- Early training of operators possible (operator training)

Local and distributed communication

Apart from the local communication via softbus, S7-PLCSIM Advanced offers a full-featured Ethernet connection and can therefore also perform distributed communication.

For the communication between STEP 7 and the instances of the PLCSIM Advanced the following communication paths and options are available:

Table 3-4: Communication paths

Communication paths	Local		Distributed
Protocol	Softbus	TCP/IP	TCP/IP
Communication interface in PLCSIM Advanced	PLCSIM	PLCSIM Virtual Ethernet Adapter	PLCSIM Virtual Ethernet Adapter
STEP 7 and instances	on a PC / a VM	on a PC / a VM	distributed

Table 3-5: Communication options

Communication options	Local		Distributed
	Softbus	TCP/IP	TCP/IP
between STEP 7 and instances	yes	yes	yes
between instances	yes	yes	yes
via OPC UA and web server	no	yes	yes
between an instance and a real hardware CPU	no	no	yes
between an instance and a real HMI V14	no	no	yes
between an instance and a simulated HMI V14	yes	yes	no

Distributed communication

Distributed communication means that S7-PLCSIM Advanced instances can be connected via the PLCSIM Virtual Switch to a network. Thus, instances can also communicate across all computers with other devices.

A communication with real or simulated CPUs or HMIs is possible.

Virtual time behavior

The virtual controller internally uses two types of clocks for the simulation:

- **Virtual clock**

Forms the basis for the application program and is used by components that are dependent on control processes (for example, cyclic OBs, cycle monitoring, minimal cycle time, system time and time calculations).

The virtual clock can be accelerated or slowed down.

- **Real clock**

Cannot be accelerated or slowed down and is used by components that are not dependent on control processes, for example, the communication with STEP 7.

With PLCSIM Advanced you can slow down or accelerate the virtual clock for test purposes.

Note

If you receive a cycle time error during the simulation of a program, slow down the virtual clock by selecting a scaling factor smaller 1.

3.1.5 Simulation

Supported CPUs

S7-PLCSIM Advanced V1.0 supports the simulation of all hardware CPUs of the S7-1500 product family. All firmware versions are supported, however, V2.0 is recommended.

3 Valuable Information

3.1 Basics of S7-PLCSIM Advanced

Table 3-6: Supported hardware

Type		Article number
Standard CPUs	CPU 1511-1 PN	6ES7511-1AK01-0AB0
	CPU 1513-1 PN	6ES7513-1AL01-0AB0
	CPU 1515-2 PN	6ES7515-2AM01-0AB0
	CPU 1516-3 PN/DP	6ES7516-3AN01-0AB0
	CPU 1517-3 PN/DP ²	6ES7517-3AP00-0AB0
	CPU 1518-4 PN/DP ²	6ES7518-4AP00-0AB0
	CPU 1518-4 PN/DP ODK ^{2, 3}	6ES7518-4AP00-3AB0
Fail-safe CPUs	CPU 1511F-1 PN	6ES7511-1FK01-0AB0
	CPU 1513F-1 PN	6ES7513-1FL01-0AB0
	CPU 1515F-2 PN	6ES7515-2FM01-0AB0
	CPU 1516F-3 PN/DP	6ES7516-3FN01-0AB0
	CPU 1517F-3 PN/DP ²	6ES7517-3FP00-0AB0
	CPU 1518F-4 PN/DP ²	6ES7518-4FP00-0AB0
	CPU 1518F-4 PN/DP ODK ^{2, 3}	6ES7518-4FP00-3AB0
Compact CPUs ¹	CPU 1511C-1 PN	6ES7511-1CK00-0AB0
	CPU 1512C-1 PN	6ES7512-1CK00-0AB0
ET 200SP-CPU's	CPU 1510SP-1 PN	6ES7510-1DJ01-0AB0
	CPU 1510SP F-1 PN	6ES7510-1SJ01-0AB0
	CPU 1512SP-1 PN	6ES7512-1DK01-0AB0
	CPU 1512SP F-1 PN	6ES7512-1SK01-0AB0
Technology CPUs	CPU 1511T-1 PN	6ES7511-1TK01-0AB0
	CPU 1515T-2 PN	6ES7515-2TM01-0AB0
	CPU 1517T-3 PN/DP ^{2, 4}	6ES7517-3TP00-0AB0
	CPU 1517TF-3 PN/DP ^{2, 4}	6ES7517-3UP00-0AB0

¹⁾ The onboard periphery within the compact CPUs is not simulated. The simulation interface corresponds to the process image.

²⁾ The ODK functionality of this CPU is not simulated.

³⁾ The simulation of this CPU only supports 5120 motion control resources.

⁴⁾ The simulation of this CPU only supports 64 cam disks.

Communication services that can be simulated

S7-PLCSIM Advanced V1.0 supports the simulation of the following communication services:

Table 3-7: Communication services that can be simulated

Communication services	Description
PG communication	On commissioning, test and diagnostic
Open communication via TCP/IP	TSEND_C / TRCV_C TSEND / TRCV TCON T_DISCON
Open communication via ISO-on-TCP	TSEND_C / TRCV_C TSEND / TRCV TCON T_DISCON
Open communication via UDP ¹	TUSEND / TURCV TCON T_DISCON
Communication via Modbus TCP	MB_CLIENT MB_SERVER
E-Mail ¹	TMAIL_C
S7 communication	PUT / GET BSEND / BRCV USEND / URCV
OPC UA Server ¹	Data exchange with OPC UA clients
Web server ¹	Data exchange via HTTPS

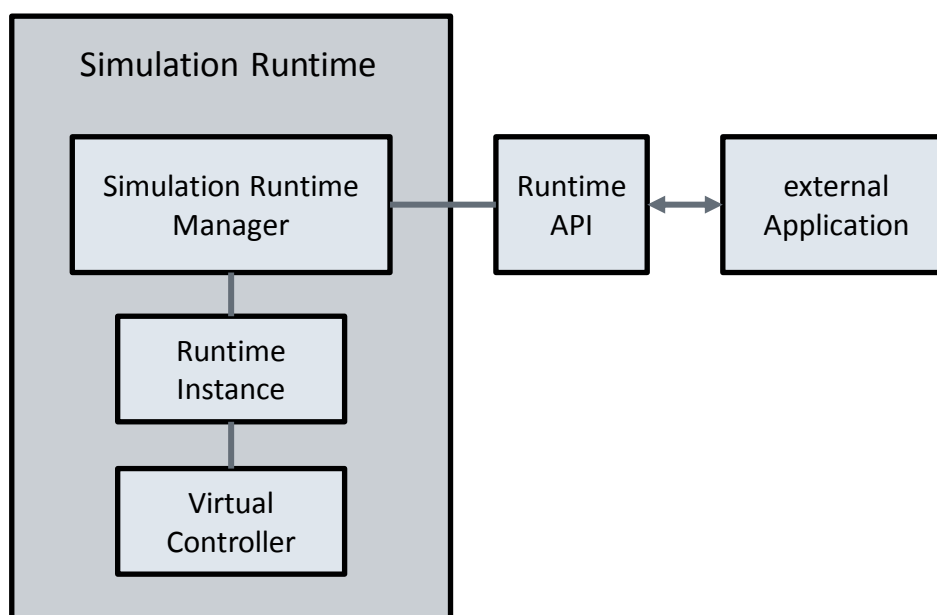
¹⁾ Only via the communication interface "PLCSIM Virtual Ethernet Adapter" (TCP/IP).

3.1.6 User interfaces (API)

Through the user interfaces PLCSIM Advanced enables the interaction with external applications, for example, own C++/C# programs or software for the simulation of production machines and systems.

You access the simulation Runtime via the Runtime API. The Simulation Runtime Manager manages the runtime instances. The runtime instances load the libraries of the virtual controller.

Figure 3-2: External application and Simulation Runtime



Components of the Simulation Runtime

The following table shows the relevant components for handling the simulation Runtime of the PLCSIM Advanced.

Table 3-8: Simulation Runtime components

Component	Description
Siemens.Simatic.Simulation.Runtime.Manager.exe	A Windows process that is running in the background. Main component of Runtime that manages all other Runtime components. The process is started automatically as soon as an application tries to initialize the Runtime API. The process is ended automatically as soon as no application is running that has initialized the Runtime API.
Siemens.Simatic.Simulation.Runtime.Instance.exe	The process of the instance that loads a DLL of a virtual controller. Every virtual controller creates its own process each.
Siemens.Simatic.Simulation.Runtime.Api.x86.dll Siemens.Simatic.Simulation.Runtime.Api.x64.dll	API libraries that has to load an application in order to use the Simulation Runtime. The libraries include interfaces for native and managed code. The "Runtime.Api.x86.dll" is exclusively loaded by 32-bit applications, the Runtime.Api.x64.dll by 64-bit applications. The DLLs are based on .NET Framework 4.0.

Component	Description
SimulationRuntimeApi.h	Header file that describes all data types that require a Native C++ application in order to use the API library.

Runtime API

The Runtime API provides the interface to an external application. These include functions with which you can, for example, create instances, change the mode of the virtual controller and exchange I/O data.

The following interfaces are used in this application example:

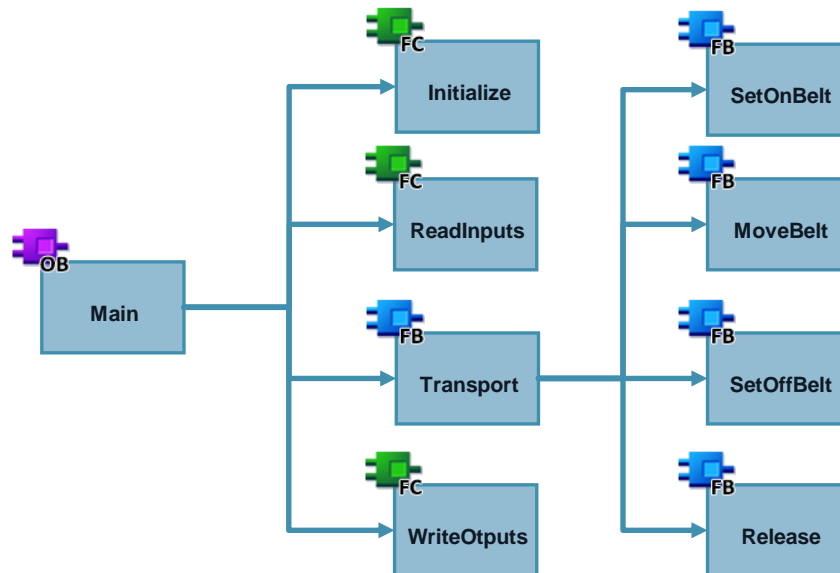
- **ISimulationRuntimeManager**
Interface of the Runtime Manager. This interface is used to register new Runtime instances, to browse already existing ones and to get an interface of a registered instance. In a Runtime Manager up to 16 instances can be registered.
- **IInstance**
Interface of a Runtime Instance. This interface is used to change the operating mode of a virtual controller and to exchange I/O data. Each instance has a unique name and an ID.

3.2 Details on the functionality

3.2.1 TIA Portal program

Structure

Figure 3-3: Configuration of the TIA Portal program



Monitoring time of the transport belt 1

The monitoring time for the transport belt 1 in the “Transport” function block is defined as a time constant with the value 5000 ms. In step 3 “Move belt” the entire activation time of the step is monitored.

Figure 3-4: Monitoring time

109739660_Transport > PLC_1 [CPU 1516-3 PN/DP] > Program blocks > Transport [FB1]

Name	Data type	Default value	Ret...	Acc...	Wr...	Vis...	Setpoint	Su...	Comment
Constant									
TIME_CHECK	Time	T#5000ms							Supervision time for step 3 (Move belt)

Navigation > Permanent pre-instruction...

Sequences (1)

1: Transport sequence

S3: Move Belt

Interlock -(c)-: Check Error

Supervision -(v)-: Check package reach belt destination position on time

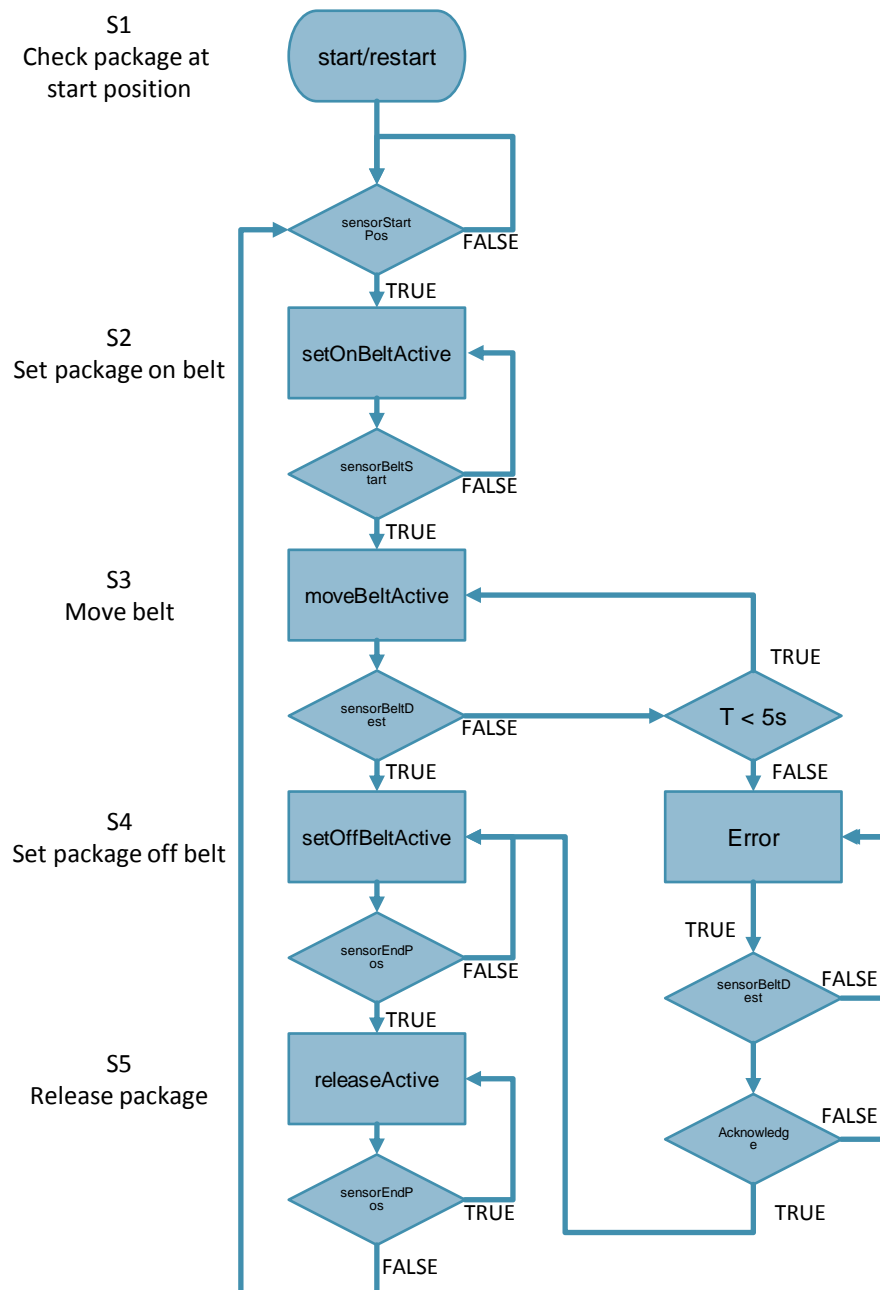
#"Move Belt".T > Time T#5000ms #TIME_CHECK

Supervision (V)

Sequence control

The complete sequence control of the conveyor system is programmed in the “Transport” function block with S7-GRAPH. The following sequence diagram provides the course of the sequence.

Figure 3-5: Sequence control of the conveyor system



3.2.2 Co-simulation program

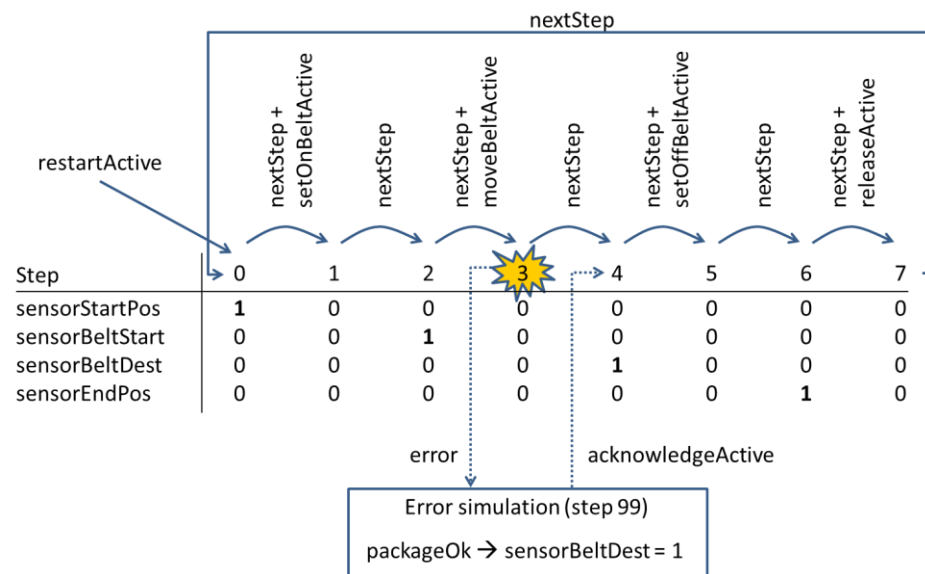
Overview

The co-simulation program is programmed in the Visual Studio project in the method "CoSimProgramm()" of the class "Cosimulation.cs".

The program is programmed in a step sequence (switch-case). For the simulation of a sensor and the operation of a machine, a timer is started in the respective step, after its lapse the "nextStep" tag is set.

The following figure shows the program sequence of the co-simulation.

Figure 3-6: Co-simulation program sequence



Error simulation

By setting the "error" tag, it is simulated that a package falls down from transport belt 1.

When the "error" tag is set, the program interrupts the regular sequence in step 3 and jumps to step 99 (error simulation).

The error simulation is ended, when the package is replaced onto transport belt 1 ("packageOk") and the error is acknowledged in the controller (acknowledgeActive) or the system is restarted (restartActive).

Note

Additional description of the co-simulation program can be found in the comments in the program code.

4 Appendix

4.1 Service and Support

Industry Online Support

Do you have any questions or need support?

Siemens Industry Online Support offers access to our entire service and support know-how as well as to our services.

Siemens Industry Online Support is the central address for information on our products, solutions and services.

Product information, manuals, downloads, FAQs and application examples – all information is accessible with just a few mouse clicks at

<https://support.industry.siemens.com/>.

Technical Support

Siemens Industry's Technical Support offers quick and competent support regarding all technical queries with numerous tailor-made offers – from basic support to individual support contracts.

Please address your requests to the Technical Support via the web form:

www.siemens.en/industry/supportrequest.

Service offer

Our service offer comprises, among other things, the following services:

- Product Training
- Plant Data Services
- Spare Parts Services
- Repair Services
- On Site and Maintenance Services
- Retrofit & Modernization Services
- Service Programs and Agreements

Detailed information on our service offer is available in the Service Catalog:

<https://support.industry.siemens.com/cs/sc>

Industry Online Support app

Thanks to the "Siemens Industry Online Support" app, you will get optimum support even when you are on the move. The app is available for Apple iOS, Android and Windows Phone.

<https://support.industry.siemens.com/cs/en/en/sc/2067>

4.2 Links and Literature

Table 4-1

No.	Topic
\1\	Industry Online Support https://support.industry.siemens.com
\2\	This document https://support.industry.siemens.com/cs/ww/en/view/109739660
\3\	SIMATIC S7-PLCSIM Advanced function manual https://support.industry.siemens.com/cs/ww/en/view/109739153

4.3 Change documentation

Table 4-2

Version	Date	Modifications
V1.0	12/2016	First version