SIEMENS

Anwendungsbeispiel • 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/de/view/109739660

Gewährleistung und Haftung

Hinweis

Die Anwendungsbeispiele sind unverbindlich und erheben keinen Anspruch auf Vollständigkeit hinsichtlich Konfiguration und Ausstattung sowie jeglicher Eventualitäten. Die Anwendungsbeispiele stellen keine kundenspezifischen Lösungen dar, sondern sollen lediglich Hilfestellung bieten bei typischen Aufgabenstellungen. Sie sind für den sachgemäßen Betrieb der beschriebenen Produkte selbst verantwortlich. Diese Anwendungsbeispiele entheben Sie nicht der Verpflichtung zu sicherem Umgang bei Anwendung, Installation, Betrieb und Wartung. Durch Nutzung dieser Anwendungsbeispiele erkennen Sie an, dass wir über die beschriebene Haftungsregelung hinaus nicht für etwaige Schäden haftbar gemacht werden können. Wir behalten uns das Recht vor, Änderungen an diesen Anwendungsbeispiele jederzeit ohne Ankündigung durchzuführen. Bei Abweichungen zwischen den Vorschlägen in diesem Anwendungsbeispiel und anderen Siemens Publikationen, wie z. B. Katalogen, hat der Inhalt der anderen Dokumentation Vorrang.

Für die in diesem Dokument enthaltenen Informationen übernehmen wir keine Gewähr.
Unsere Haftung, gleich aus welchem Rechtsgrund, für durch die Verwendung der in diesem Anwendungsbeispiel beschriebenen Beispiele, Hinweise, Programme, Projektierungs- und Leistungsdaten usw. verursachte Schäden ist ausgeschlossen, soweit nicht z. B. nach dem Produkthaftungsgesetz in Fällen des Vorsatzes, der groben Fahrlässigkeit, wegen der Verletzung des Lebens, des Körpers oder der Gesundheit, wegen einer Übernahme der Garantie für die Beschaffenheit einer Sache, wegen des arglistigen Verschweigens eines Mangels oder wegen Verletzung wesentlicher Vertragspflichten zwingend gehaftet wird. Der Schadensersatz wegen Verletzung wesentlicher Vertragspflichten ist jedoch auf den vertragstypischen, vorhersehbaren Schaden begrenzt, soweit nicht Vorsatz oder grobe Fahrlässigkeit vorliegt oder wegen der Verletzung des Lebens, des Körpers oder der Gesundheit zwingend gehaftet wird. Eine Änderung der Beweislast zu Ihrem Nachteil ist hiermit nicht verbunden.

Weitergabe oder Vervielfältigung dieser Anwendungsbeispiele oder Auszüge daraus sind nicht gestattet, soweit nicht ausdrücklich von der Siemens AG zugestanden.

Security-hinweise

Siemens bietet Produkte und Lösungen mit Industrial Security-Funktionen an, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen.

Um Anlagen, Systeme, Maschinen und Netzwerke gegen Cyber-Bedrohungen zu sichern, ist es erforderlich, ein ganzheitliches Industrial Security-Konzept zu implementieren (und kontinuierlich aufrechtzuerhalten), das dem aktuellen Stand der Technik entspricht. Die Produkte und Lösungen von Siemens formen nur einen Bestandteil eines solchen Konzepts.

Der Kunde ist dafür verantwortlich, unbefugten Zugriff auf seine Anlagen, Systeme, Maschinen und Netzwerke zu verhindern. Systeme, Maschinen und Komponenten sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn und soweit dies notwendig ist und entsprechende Schutzmaßnahmen (z.B. Nutzung von Firewalls und Netzwerksegmentierung) ergriffen wurden.

Zusätzlich sollten die Empfehlungen von Siemens zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Industrial Security finden Sie unter http://www.siemens.com/industrialsecurity.

Die Produkte und Lösungen von Siemens werden ständig weiterentwickelt, um sie noch sicherer zu machen. Siemens empfiehlt ausdrücklich, Aktualisierungen durchzuführen, sobald die entsprechenden Updates zur Verfügung stehen und immer nur die aktuellen Produktversionen zu verwenden. Die Verwendung veralteter oder nicht mehr unterstützter Versionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Produkt-Updates informiert zu sein, abonnieren Sie den Siemens Industrial Security RSS Feed unter http://www.siemens.com/industrialsecurity.

Inhaltsverzeichnis

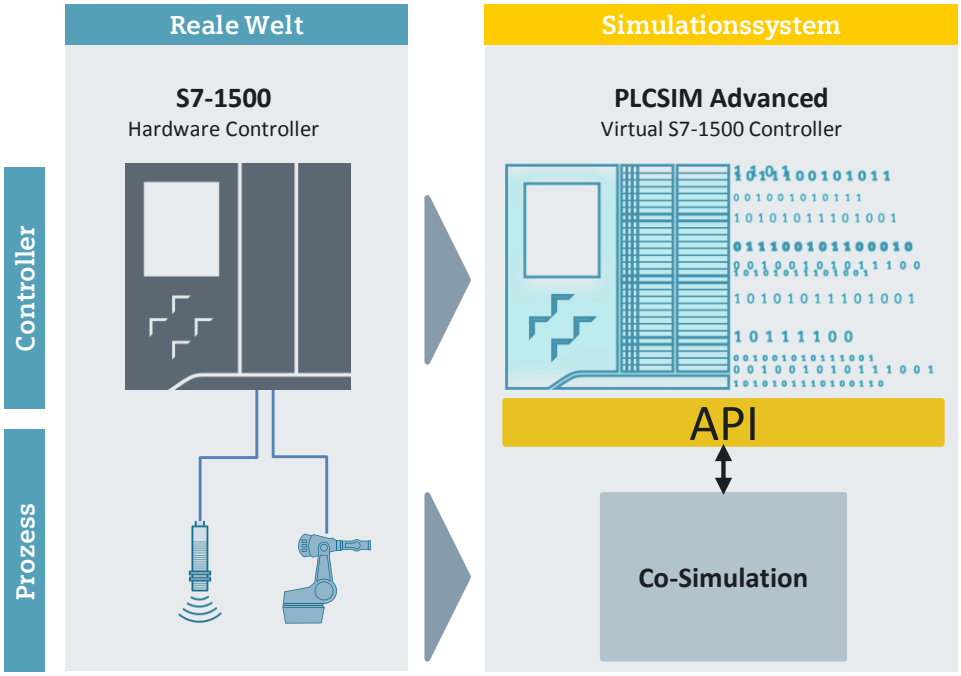
1 Einführung

1.1 Überblick

Mit SIMATIC S7-PLCSIM Advanced werden virtuelle Controller zur Simulation einer S7-1500 oder ET 200SP CPU erstellt und zur umfangreichen Funktionssimulation genutzt. Um ein STEP 7-Programm zu testen, werden demnach keine realen Controller benötigt.

Die virtuellen Controller können auch im Kontext einer Anlage oder Maschine getestet und validiert werden. Um den virtuellen Controller an eine Anlagen- oder Maschinen-Simulation (Co-Simulation) anzubinden, wird eine Anwenderschnittstelle (API) verwendet.

Abbildung 1-1: Überblick



Vorteile des Anwendungsbeispiels

- Einstieg in die Verwendung der API
- C#-Beispielcode, auf dem Sie eigene Anwendungen aufbauen können

Abgrenzung

Dieses Anwendungsbeispiel enthält keine Beschreibung zu folgenden Themen:

- Grundlagen der objektorientierten Programmierung
- Grundlagen der Programmierumgebung, z. B. Microsoft Visual Studio
- Grundlagen der TIA Portal-Projektierung

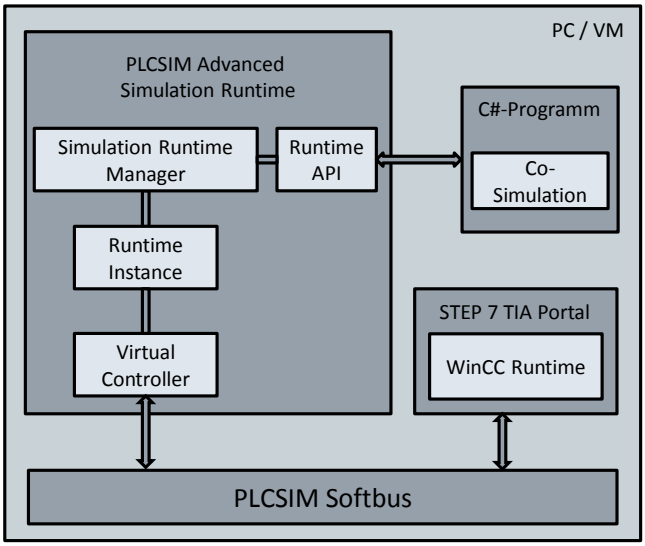
Ausreichende Kenntnisse über diese Themen werden vorausgesetzt.

1.2 Funktionsweise

1.2.1 Übersicht

Folgende Abbildung stellt eine Übersicht über das Funktionsprinzip des Anwendungsbeispiels dar.

Abbildung 1-2: Übersicht der Systeme



PLCSIM Advanced

Mit S7-PLCSIM Advanced als eigenständiges Simulationssystem kann das erstellte Anwenderprogramm auf einem virtuellen Controller simuliert und getestet werden.

Die Runtime API von S7-PLCSIM Advanced (Details siehe Kapitel 3.1.6) stellt Anwenderschnittstellen für den Zugriff auf die Simulation Runtime bereit. Diese Anwenderschnittstellen bieten folgende Möglichkeiten über ein Programm (z. B. C#-Programm):

- Erzeugen einer Runtime Instanz eines virtuellen Controllers.
- Ändern des Betriebszustands des virtuellen Controllers.
- Austausch von I/O-Daten mit einer Co-Simulation.

In diesem Anwendungsbeispiel wird eine lokale Kommunikation über den PLCSIM Softbus verwendet. Mit S7-PLCSIM Advanced ist aber auch eine verteilte Kommunikation über einen virtuellen Ethernet Adapter möglich.

STEP 7 (TIA Portal) Programm

Ein in STEP 7 V14 erstelltes STEP 7-Programm (Details siehe Kapitel 3.2.1) steuert eine Transportanlage. Für einen umfangreichen Funktionstest wird das STEP 7-Programm über S7-PLCSIM Advanced in einen virtuellen S7-1500-Controller geladen. Dieser Controller interagiert über die API mit einer Co-Simulation (Anlagensimulation), um im Kontext der Anlage das STEP 7-Programm zu validieren.

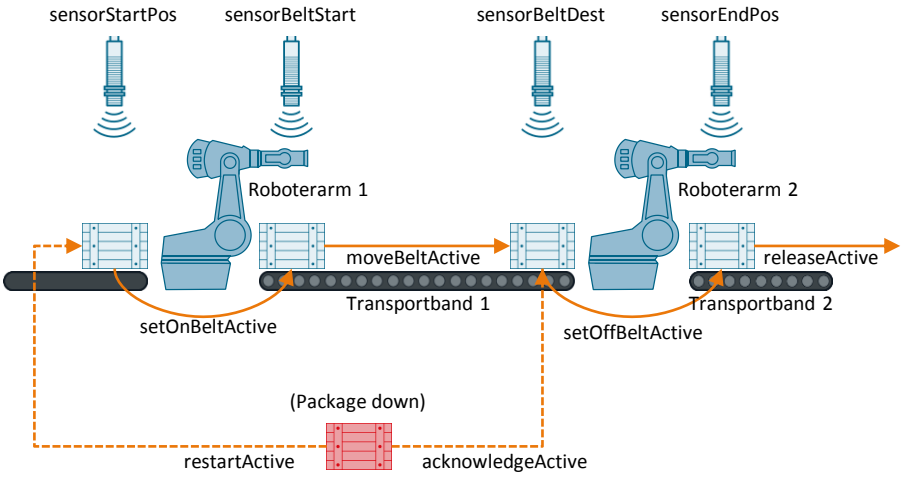
Die zu steuernde Transportanlage besteht aus zwei Transportbändern und zwei Roboterarmen. Damit der Controller mit der Transportanlage kommuniziert, werden folgende Ein- und Ausgangsvariablen definiert:

Tabelle 1-1: Ein- und Ausgangsvariablen

Variable	Datentyp	Typ	Beschreibung
sensorStartPos	BOOL	Eingang	Sensorsignal: Startposition
sensorBeltStartPos	BOOL	Eingang	Sensorsignal: Startposition Transportband 1
sensorBeltDest	BOOL	Eingang	Sensorsignal: Zielposition Transportband 1
sensorEndPos	BOOL	Eingang	Sensorsignal: Endposition
setOnBeltActive	BOOL	Ausgang	Steuersignal: Roboterarm 1 aktivieren
moveBeltActive	BOOL	Ausgang	Steuersignal: Transportband 1 aktivieren
setOffBeltActive	BOOL	Ausgang	Steuersignal: Roboterarm 2 aktivieren
releaseActive	BOOL	Ausgang	Steuersignal: Transportband 2 aktivieren
acknowledgeActive	BOOL	Ausgang	Steuersignal: Fehler quittieren
restartActive	BOOL	Ausgang	Steuersignal: Anlage neustarten

Ausgehend von den aktiven Sensorsignalen wird das entsprechende Steuersignal aktiviert. Für das Transportband 1 (moveBeltActive) ist eine Überwachungszeit von 5 s projektiert. Erreicht die Kiste innerhalb dieser Zeit die Zielposition nicht, wird ein Fehler für eine heruntergefallene Kiste angezeigt („Package down“). Sie können dann die Anlage neu starten oder den Fehler quittieren, wenn Sie die Kiste wieder zurück auf das Transportband 1 stellen.

Abbildung 1-3: Übersicht Transportanlage



Die Visualisierung der Anlage wird mit der WinCC Runtime simuliert. In der WinCC Runtime kann die Anlage über ein HMI-Bild beobachtet und gesteuert werden:

- Neustart (restartActive)
- Fehler quittieren (acknowledgeActive)

C#-Programm

In der externen Anwendung (C#-Programm) wird folgendes programmiert:

- Funktionen der Anwenderschnittstelle (Runtime API), um mit dem virtuellen Controller von S7-PLCSIM Advanced zu interagieren.
- Anlagensimulation der Transportanlage (Co-Simulation).

Die Co-Simulation reagiert auf die Steuersignale (Ausgänge) des virtuellen Controllers und simuliert alle notwendigen Sensorsignale (Eingänge) für die Simulation der Ablaufsteuerung (Details siehe Kapitel 3.2.2).

Für den Austausch der I/O-Daten schreibt und liest die Runtime API aus einem Speicherbereich, der am Zykluskontrollpunkt mit dem internen Prozessabbild des virtuellen S7-1500 Controllers synchronisiert wird.

Tabelle 1-2: Datenaustausch

Virtueller Controller	Datenaustausch	Co-Simulation
setOnBeltActive	→	setOnBeltActive
moveBeltActive	→	moveBeltActive
setOffBeltActive	→	setOffBeltActive
releaseActive	→	releaseActive
acknowledgeActive	→	acknowledgeActive
restartActive	→	restartActive
sensorStartPos	←	sensorStartPos
sensorBeltStart	←	sensorBeltStart
sensorBeltDest	←	sensorBeltDest
sensorEndPos	←	sensorEndPos

Mit der Co-Simulation können Sie Fehler simulieren bzw. beheben, z. B. eine Kiste herunterfallen lassen bzw. eine heruntergefallene Kiste wieder auf das Transportband 1 zurücksetzen.

Hinweis Die Maschinen der Transportanlage werden in der Co-Simulation nicht realitätsnah nachgebildet. Die Bewegungen werden lediglich durch eine Zeitkonstante simuliert.

1.2.2 Funktionsumfang

In diesem Anwendungsbeispiel werden folgende Funktionen der Runtime API verwendet und dargestellt.

Tabelle 1-3: API-Funktionen

Funktion	Beschreibung	Siehe
`RegisterInstance()`	Registriert im Runtime Manager eine neue Instanz eines virtuellen Controllers.	S. 13
`UnregisterInstance()`	Meldet eine Instanz vom Runtime Manager ab.	S. 13
`PowerOn()`	Erzeugt den Prozess für die Simulation Runtime Instanz und startet die Firmware des virtuellen Controllers.	S. 14
`PowerOff()`	Fährt die Simulation Runtime Instanz herunter und schließt deren Prozess.	S. 15
`Run()`	Fordert vom virtuellen Controller, in den Betriebszustand RUN zu wechseln.	S. 15
`Stop()`	Fordert vom virtuellen Controller, in den Betriebszustand STOP zu wechseln.	S. 16
`SetIPSuite()`	Setzt die IP-Suite der Netzwerk-Schnittstelle eines virtuellen Controllers.	S. 14
`UpdateTagList()`	Liest die Variablen aus dem virtuellen Controller und schreibt sie nach Namen geordnet in den gemeinsamen Speicher.	S. 18
`ReadBool()`	Liest symbolisch den Wert einer PLC-Variablen.	S. 18
`WriteBool()`	Schreibt symbolisch den Wert einer PLC-Variablen.	S. 18
`IsAlwaysSendOnEndOfCycleEnabled{get; set;}`	Liefert oder setzt den Modus `AlwaysSendOnEndOfCycle`. Wenn der Modus gesetzt ist, wird für jede Betriebsart nach jedem Zyklusende das Ereignis `OnEndOfCycle` ausgelöst.	S.13
`CommunicationInterface {get; set;}`	Setzt die Kommunikations-Schnittstelle des virtuellen Controllers oder liefert sie zurück: Lokale Kommunikation (Softbus) oder TCP/IP.	S.13
`OnConfigurationChanged`	Dieses Ereignis wird ausgelöst, wenn sich die Konfiguration des virtuellen Controllers geändert hat.	S.13
`OnEndOfCycle`	Dieses Ereignis wird ausgelöst, wenn der virtuelle Controller das Ende des Hauptzyklus erreicht.	S.13

1.3 Verwendete Komponenten

Hardware

Hinweis In diesem Anwendungsbeispiel wurde keine reale Hardware eingesetzt. Alle projektierten Hardware-Komponenten wurden simuliert.

Tabelle 1-4: Hardware-Komponenten

Komponente	Anzahl	Artikelnummer	Hinweis
CPU 1516-3 PN/DP	1	6ES7516-3AN01-0AB0	V2.0
TP900 Comfort	1	6AV2124-0JC01-0AX0	V14.0.0.0

Software

Tabelle 1-5: Software-Komponenten

Komponente	Anzahl	Artikelnummer
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

Tabelle 1-6: Downloads

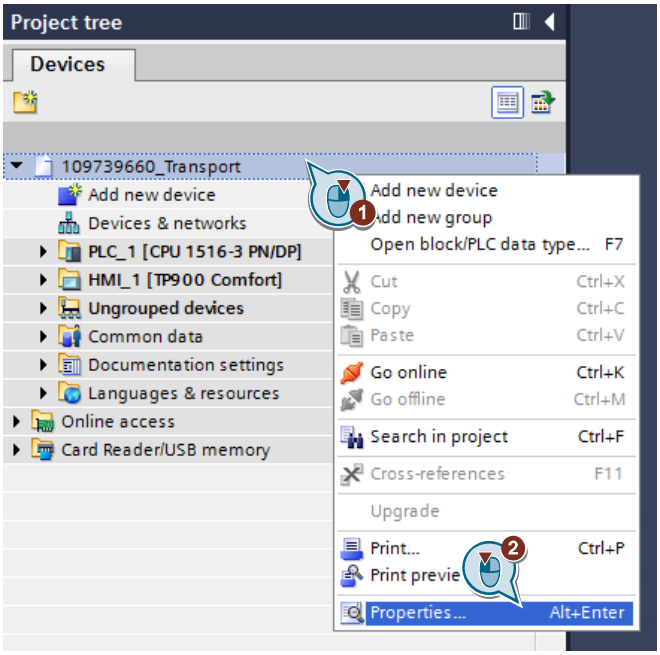
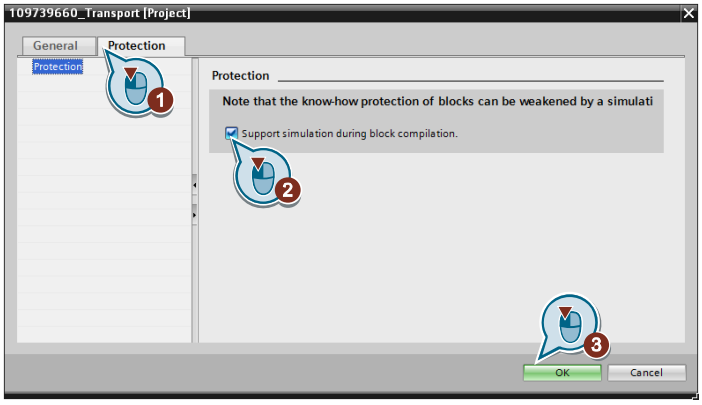
Datei	Inhalt
109739660_PLCSIM_Advanced_DOC_V10_de.pdf	Dieses Dokument
109739660_PLCSIM_Advanced_S7_PROJ_V10.zip	TIA Portal-Projekt (Anlagensteuerung)
109739660_PLCSIM_Advanced_COSIM_APPL_V10.zip	Anlagensimulation (Anwendung und Visual Studio Projekt)

2 Engineering

2.1 Projektierung im TIA Portal

Das mitgelieferte TIA Portal-Projekt bedarf keiner weiteren Konfiguration. Wenn Sie ein eigenes Projekt erstellen, führen Sie folgende Einstellungen durch, um mit S7-PLCSIM Advanced zu simulieren.

Tabelle 2-1: Einstellung in TIA Portal

Nr.	Aktion
1.	1. Öffnen Sie mit einem Rechtsklick das Kontextmenü für das Projekt. 2. Klicken Sie auf „Eigenschaften…“ („Properties…“). 
2.	1. Wählen Sie das Register „Schutz“ („Protection“) aus. 2. Aktivieren Sie das Optionskästchen „Beim Übersetzen von Bausteinen Simulierbarkeit unterstützen.“ („Support simulation during block compilation.“). 3. Bestätigen Sie mit „OK“. 

2.2 Programmierung in Visual Studio

Das mitgelieferte Visual Studio-Projekt erleichtert Ihnen den Einstieg zur Programmierung einer S7-PLCSIM Advanced-Anwendung mit einer Co-Simulation.

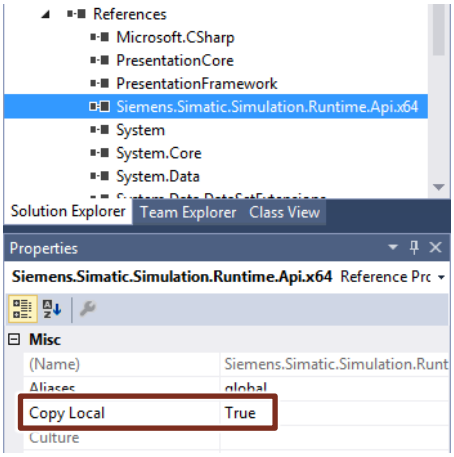
In dem Visual Studio-Projekt sind einige grundlegende Funktionen bereits programmiert, z. B. virtuellen Controller einschalten und I/O-Daten austauschen. Mit diesem Visual Studio-Projekt können Sie Ihre Anwendungen weiterentwickeln.

In den Kapitel 2.2.2 bis 2.2.4 finden Sie einen Überblick über die C#- und API-Funktionen, die in diesem Anwendungsbeispiel programmiert sind.

2.2.1 API-Bibliothek einbinden

Hinweis Damit im mitgelieferten Projekt der Pfad aktualisiert wird, müssen Sie die vorhandene Referenz auf die Bibliothek „Siemens.Simatic.Simulation.Runtime.Api.x64“ löschen und erneut einbinden.

Tabelle 2-2: API-Bibliothek einbinden

Nr.	Aktion
1.	Legen Sie im Projekt eine Referenz zu der S7-PLCSIM Advanced Bibliothek an. Diese befindet sich im folgenden Verzeichnis unter „…\Program Files (x86)\Common Files\Siemens\PLCSIMADV\API“.
2.	Stellen Sie die Eigenschaft „Lokale Kopie“ („Copy Local“) der Referenz auf „True“. 

2.2.2 Übersicht Bedienoberfläche

In der graphischen Bedienoberfläche (GUI, engl. graphical user interface) der externen Anwendung werden über die Schaltflächen (1) Kommandos eingebunden, um mit dem virtuellen Controller (PLC Instance) und der Co-Simulation zu interagieren.

Das Listenfeld (2) zeigt Infos und Fehlermeldungen an. Das Textfeld (3) zeigt den jeweiligen Status an.

Abbildung 2-1: GUI

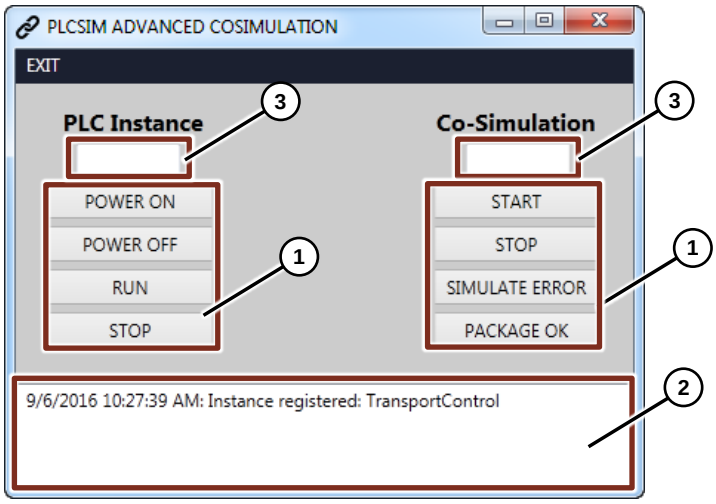
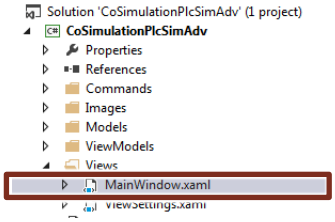
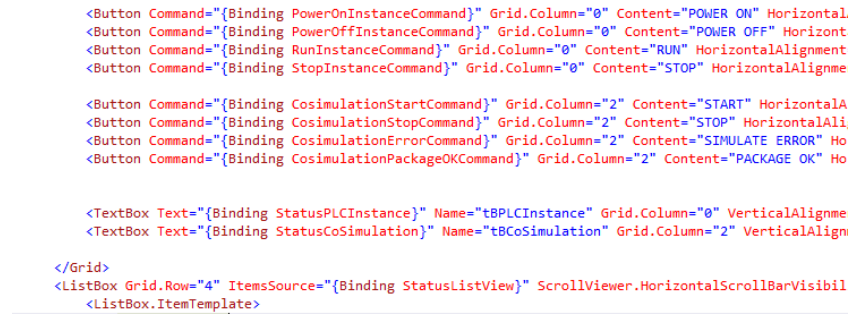


Tabelle 2-3: Kommandos

Nr.	Beschreibung
1.	Öffnen Sie das Projekt „CoSimulationPlcSimAdv“ aus dem beigefügten Download „109739660_PLCSIM_Advanced_COSIM_APPL_V10.zip“.
2.	Öffnen Sie die Ansicht des Hauptfensters „MainWindow.xaml“ im Projektmappen-Explorer in dem Ordner „Views“.  Hier finden Sie im XAML-Code unter anderem die eingebundenen Kommandos für die Schaltflächen bzw. die eingebundenen Daten für die Anzeigefelder. 


```
<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" HorizontalAlignment
<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>
```

2.2.3 Programmierung der PLC Instance

Instanz erzeugen

Tabelle 2-4: Klasse „MainWindowViewModel.cs“

Nr.	Beschreibung
1.	Öffnen Sie in dem Ordner „ViewModels“ die Klasse „MainWindowViewModel.cs“.

2.	Im #region-Block „Fields“ ist eine Variable „virtualController“ vom Objekttyp „PLCInstance“ definiert.
	`/// <summary>`
	`/// PLCSIM Adv. Instance of the virtual controller`
	`/// </summary>`
	`public PLCInstance virtualController = null;`

3.	Im Konstruktor der Klasse (#region-Block „C’Tor“) wird ein neues Objekt der Klasse „PLCInstance“ angelegt und der Variablen „virtualController“ zugewiesen. Als Name für die neue Instanz wird „TransportControl“ übergeben.
	`public MainWindowViewModel()`
	`{`
	`StatusListView = new ObservableCollection<String>();`
	`try`
	`{`
	`virtualController = new PLCInstance("TransportControl");`

Tabelle 2-5: Klasse „PLCInstance.cs“

Nr.	Beschreibung
1.	Öffnen Sie in dem Ordner „Models“ die Klasse „PLCInstance.cs“.

2.	Im #region-Block „Properties“ ist eine Variable „instance“ als Anwenderschnittstelle „IInstance“ definiert.
	`/// <summary>`
	`/// instance of PLCSIM Adv. virtual Controller`
	`/// </summary>`
	`public IInstance instance { get; set; }`

3.	Im Konstruktor der Klasse (#region-Block „C’Tor“) wird mit der Funktion „RegisterInstance()“ der API ISimulationRuntimeManager eine neue Instanz eines virtuellen Controllers im Runtime Manager registriert. Die Funktion erzeugt und liefert eine Schnittstelle von dieser Instanz zurück.
	Die erzeugte Schnittstelle wird der Variablen „instance“ zugewiesen.
	`public PLCInstance(string instanceName)`
	`{`
	`instance = SimulationRuntimeManager.RegisterInstance(instanceName);`
	`instance.IsAlwaysSendOnEndOfCycleEnabled = true;`
	`instance.CommunicationInterface = ECommunicationInterface.Softbus;`
	`instance.OnConfigurationChanged += instance_OnConfigurationChanged;`
	`instance.OnEndOfCycle += instance_OnEndOfCycle;`
	`}`

4.	Zusätzlich werden für die registrierte Instanz folgende Einstellungen vorgenommen: - „instance.CommunicationInterface = ECommunicationInterface.Softbus“ stellt die Kommunikations-Schnittstelle des virtuellen Controllers auf lokale Kommunikation (Softbus) ein. - „instance.IsAlwaysSendOnEndOfCycleEnabled = true“ bewirkt, dass am Ende jedes CPU-Zyklus das Event „OnEndOfCycle“ ausgelöst wird. - Den Events „OnConfigurationChanged“ und „OnEndOfCycle” wird jeweils ein Eventhandler zugewiesen.

Nr.	Beschreibung

	public PLCInstance(string instanceName)
	{
	instance = SimulationRuntimeManager.RegisterInstance(instanceName);
	instance.IsAlwaysSendOnEndOfCycleEnabled = true;
	instance.CommunicationInterface = ECommunicationInterface.Softbus;
	instance.OnConfigurationChanged += instance_OnConfigurationChanged;
	instance.OnEndOfCycle += instance_OnEndOfCycle;
	}

Instanz abmelden

Tabelle 2-6: Klasse „MainWindow.xaml.cs“

Nr.	Beschreibung
1.	Öffnen Sie in dem Ordner „Views“ die Klasse „MainWindow.xaml.cs“.

2.	Durch das Ereignis „Window_Closing“ wird die Funktion „UnregisterInstance()“ der API IInstance aufgerufen. Diese meldet die Instanz des virtuellen Controllers wieder vom Runtime Manager ab.
	private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
	{
	ViewModels.MainWindowViewModel Application = this.DataContext as ViewModels.MainWindowViewModel;
	//Unregister Instance
	Application.virtualController.instance.UnregisterInstance();
	}

Die Anwendung wird über die Schaltfläche „EXIT“ oder das „Schließen“-Symbol der Bedienoberfläche geschlossen. Dabei wird das Ereignis „Window_Closing“ ausgelöst.

Instanz einschalten

Tabelle 2-7: Klasse „MainWindowViewModel.cs“

Nr.	Beschreibung
1.	Öffnen Sie in dem Ordner „ViewModels“ die Klasse „MainWindowViewModel.cs“.

2.	Durch Betätigen der Schaltfläche „POWER ON“ auf der Bedienoberfläche wird im #region-Block „Public Methods“ die Methode „PowerOnPLCInstance()“ des virtuellen Controllers aufgerufen.
	/// <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));
	}
	}

Tabelle 2-8: Klasse „PLCInstance.cs“

Nr.	Beschreibung
1.	Öffnen Sie in dem Ordner „Models“ die Klasse „PLCInstance.cs“.
2.	Die Methode „PowerOnPLCInstance()“ im #region-Block „Public Methods“ ruft die Funktionen „PowerOn()“ und „SetIPSuite()“ der API IInstance auf. Die Funktion „PowerOn()“ erzeugt den Prozess für die Simulation Runtime Instanz und startet die Firmware des virtuellen Controllers. Als Timeout-Wert sind 60.000 ms parametriert.

Nr.	Beschreibung

	Die Funktion „SetIPSuite()“ konfiguriert die Netzwerk-Schnittstelle des virtuellen Controllers.
	`/// <summary>`
	`/// Power On PLCSIM Adv. Instanz, set IPSuite of instance`
	`/// </summary>`
	`/// <returns></returns>`
	`public void PowerOnPLCInstance()`
	`{`
	`instance.PowerOn(60000);`
	`instance.SetIPSuite(0, instanceIP, true);`
	`}`

3.	Die Parameter (IP-Adresse, Subnetzmaske und Standard-Gateway) der Netzwerk-Schnittstelle sind im #region-Block „Properties“ in der Variablen „instanceIP“ vom Strukturtyp „SIPSuite4“ definiert. `private SIPSuite4 instanceIP = new SIPSuite4("192.168.0.101", "255.255.255.0", "0.0.0.0");`

Instanz ausschalten

Tabelle 2-9: Klasse „MainWindowViewModel.cs“

Nr.	Beschreibung
1.	Öffnen Sie in dem Ordner „ViewModels“ die Klasse „MainWindowViewModel.cs“.

2.	Durch Betätigen der Schaltfläche „POWER OFF“ auf der Bedienoberfläche wird im #region-Block „Public Methods“ die Methode „PowerOffPLCInstance()“ des virtuellen Controllers aufgerufen.
	`/// <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 (SimulationRuntimeException simRtEx)`
	`{`
	`WriteStatusEntry(String.Format("PowerOff Instance failed: {0}", simRtEx.Message));`
	`}`
	`}`

Tabelle 2-10: Klasse „PLCInstance.cs“

Nr.	Beschreibung
1.	Öffnen Sie in dem Ordner „Models“ die Klasse „PLCInstance.cs“.

2.	Die Methode „PowerOffPLCInstance()“ im #region-Block „Public Methods“ ruft die Funktion „PoweOff()“ der API IInstance auf. Diese fährt die Simulation Runtime herunter und schließt deren Prozess. Als Timeout-Wert sind 6.000 ms parametriert.
	`/// <summary>`
	`/// Power Off PLCSIM Adv. Instance`
	`/// </summary>`
	`public void PowerOffPLCInstance()`
	`{`
	`instance.PowerOff(6000);`
	`}`

Instanz starten

Tabelle 2-11: Klasse „MainWindowViewModel.cs“

Nr.	Beschreibung
1.	Öffnen Sie in dem Ordner „ViewModels“ die Klasse „MainWindowViewModel.cs“.

2.	Durch Betätigen der Schaltfläche „RUN“ auf der Bedienoberfläche wird im #region-Block „Public Methods“ die Methode „RunPLCInstance()“ des virtuellen Controllers aufgerufen.
	`/// <summary>`
	`/// Run registred Instance of virtual controller`
	`/// </summary>`
	`public void RunController()`
	`{`
	`try`
	`{`
	`virtualController.RunPLCInstance();`
	`}`
	`catch (SimulationRuntimeException simRtEx)`
	`{`
	`WriteStatusEntry(String.Format("Run Instance failed: {0}! Please load plc program before execute RUN.", simRtEx.Message));`
	`}`
	`}`

Tabelle 2-12: Klasse „PLCInstance.cs“

Nr.	Beschreibung
1.	Öffnen Sie in dem Ordner „Models“ die Klasse „PLCInstance.cs“.

2.	Die Methode „RunPLCInstance()“ im #region-Block „Public Methods“ ruft die Funktion „Run()“ der API IInstance auf. Diese fordert vom virtuellen Controller, in den Betriebszustand RUN zu wechseln. Als Timeout-Wert sind 6.000 ms parametriert.
	`/// <summary>`
	`/// Run PLCSIM Adv. Instance`
	`/// </summary>`
	`public void RunPLCInstance()`
	`{`
	`instance.Run(6000);`
	`}`

Instanz stoppen

Tabelle 2-13: Klasse „MainWindowViewModel.cs“

Nr.	Beschreibung
1.	Öffnen Sie in dem Ordner „ViewModels“ die Klasse „MainWindowViewModel.cs“.

2.	Durch Betätigen der Schaltfläche „STOP“ auf der Bedienoberfläche wird im #region-Block „Public Methods“ die Methode „StopPLCInstance()“ des virtuellen Controllers aufgerufen.
	`/// <summary>`
	`/// Stop registred Instance of virtual controller`
	`/// </summary>`
	`public void StopController()`
	`{`
	`  try`
	`  {`
	`  WriteStatusEntry(String.Format("Stop Instance: {0}", virtualController.instance.Name));`
	`  virtualController.StopPLCInstance();`
	`  }`
	`  catch (SimulationRuntimeException simRtEx)`
	`  {`
	`  WriteStatusEntry(String.Format("Stop Instance failed: {0}", simRtEx.Message));`
	`  }`
	`}`

Tabelle 2-14: Klasse „PLCInstance.cs“

Nr.	Beschreibung
1.	Öffnen Sie in dem Ordner „Models“ die Klasse „PLCInstance.cs“.

2.	Die Methode „StopPLCInstance()“ im #region-Block „Public Methods“ ruft die Funktion „Stop()“ der API IInstance auf. Diese fordert vom virtuellen Controller, in den Betriebszustand STOP zu wechseln. Als Timeout-Wert wurden 6.000 ms parametriert.
	`/// <summary>`
	`/// Stop PLCSIM Adv. Instance`
	`/// </summary>`
	`public void StopPLCInstance()`
	`{`
	`  instance.Stop(6000);`
	`}`
	
	`#endregion //Public Methods`

I/O-Daten austauschen

Tabelle 2-15: Klasse „PLCInstance.cs“

Nr.	Beschreibung
1.	Öffnen Sie in dem Ordner „Models“ die Klasse „PLCInstance.cs“.

2.	Der Eventhandler „instance_OnConfigurationChanged“ im #region-Block „Events“ ruft die Funktion „UpdateTagList()“ der API IInstance auf. Diese liest die Variablen aus dem virtuellen Controller und schreibt sie nach Namen geordnet in den gemeinsamen Speicher.
	Zum Auslesen der Ein- und Ausgangsvariablen des virtuellen Controllers wird der Funktion der Parameter „ETagListDetails.IO“ übergeben.
	`/// <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,`
	`IInstanceConfigChanged, uint in_Param1, uint in_Param2, uint in_Param3, uint in_Param4)`
	`{`
	`IsConfigured = false;`
	`try`
	`{`
	`instance.UpdateTagList(ETagListDetails.IO);`
	`IsConfigured = true;`
	`}`
	`catch (Exception ex)`
	`{`
	`}`
	`}`

3.	Der Eventhandler „instance_OnEndOfCycle“ im #region-Block „Events“ ruft die Funktionen „ReadBool()“ und „WriteBool()“ der API IInstance auf. Dadurch werden boolsche Variablen des virtuellen Controllers über den Variablennamen gelesen bzw. geschrieben.
	`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)`
	`{`
	`}`
	`}`
	`}`

2.2.4 Programmierung der Co-Simulation

Simulationsparameter

Tabelle 2-16: Klasse „MainWindowViewModel.cs“

Nr.	Beschreibung
1.	Öffnen Sie in dem Ordner „ViewModels“ die Klasse „MainWindowViewModel.cs“.
2.	Im #region-Block „Fields“ ist eine Variable „transportSystem“ vom Objekttyp „Cosimulation“ definiert. `public Cosimulation transportSystem = null;`

3.	Im Konstruktor der Klasse (#region-Block „C’tor“) wird ein neues Objekt „transportSystem“ der Klasse „Cosimulation“ angelegt.
	Simulationsparameter:
	- Simulationszeit der Roboterarme und Antriebe: 2.000 ms
	- Simulationszeit der Sensoren: 1.000 ms
	`// New Cosimulation "transportSystem`
	`// Simulation time movement 2000ms`
	`// Simulation time sensor 1000ms`
	`transportSystem = new Cosimulation(2000, 1000);`

Datenaustausch

Tabelle 2-17: Klasse „MainWindowViewModel.cs“

Nr.	Beschreibung
1.	Öffnen Sie in dem Ordner „ViewModels“ die Klasse „MainWindowViewModel.cs“.
2.	Im Konstruktor der Klasse (#region-Block „C’tor“) ist ein Verweis des Objektes „coSimulation“ des virtuellen Controllers auf das Objekt „transportSystem“ hinterlegt. `virtualController.coSimulation = transportSystem;`

Tabelle 2-18: Klasse „PLCInstance.cs“

Nr.	Beschreibung
1.	Öffnen Sie in dem Ordner „Models“ die Klasse „PLCInstance.cs“.

2.	Im Eventhandler „instance_OnEndOfCycle“ im #region-Block „Events“ werden die gelesenen Ausgangsvariablen des virtuellen Controllers an die Co-Simulation übergeben (1). Darauf erfolgt der Aufruf des Co-Simulations-Programms „CoSimProgramm()“ (2). Abschließend werden die simulierten Werte der Co-Simulation auf die Eingangsvariablen des virtuellen Controllers geschrieben (3).
	`// 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();` (2)
	(1)
	(3)
	`// 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);`

Simulationsprogramm

Tabelle 2-19: Klasse „Cosimulation.cs“

Nr.	Beschreibung
1.	Öffnen Sie in dem Ordner „Models“ die Klasse „Cosimulation.cs“.

2.	In der Deklaration der Klasse sind boolesche Variablen für den Datenaustausch mit dem virtuellen Controller definiert.
	`//--// 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;`
	`//--//`

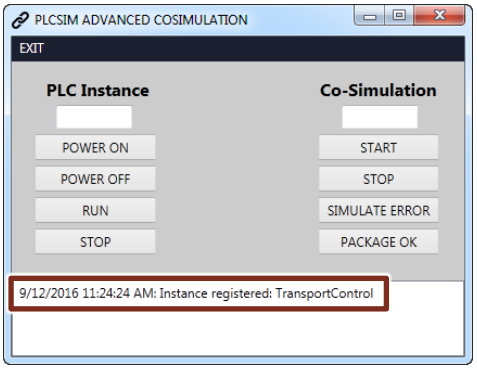
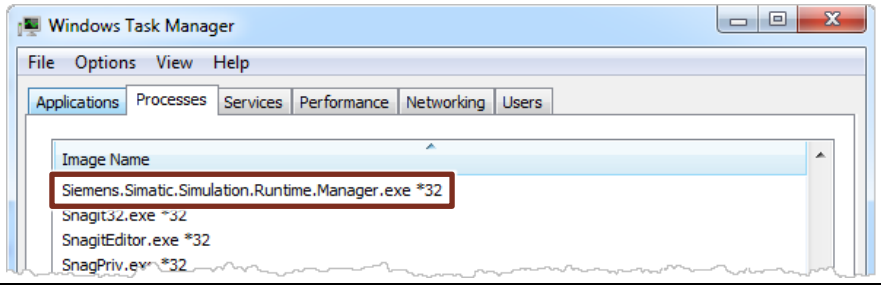
3.	In der Methode „CoSimProgramm()“ im #region-Block „Cosimulation“ sind die Zustände der simulierten Anlage in einer Switch-Case-Anweisung programmiert. Weitere Beschreibung finden Sie im Kapitel 3.2.2 oder in den Code-Kommentaren.
	`public void CoSimProgramm()`
	`{`
	`  if (restartActive) // Restart command from the virtual controller`
	`  {`
	`  step = 0; // Set start step`
	`  startedTimer = false; // Reset indicator for started timer`
	`  OnErrorSimulationStateChanged(this, new PropertyArgs("Black"));  // Reset "SIMULATE ERROR" button collor`
	`  }`
	
	`  if (run) // Active when "START" button pressed for Co-Simulation`
	`  {`
	`  OnOperatingStateChanged(this, new PropertyArgs("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;`

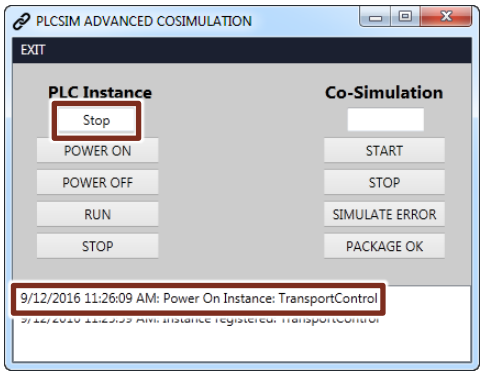
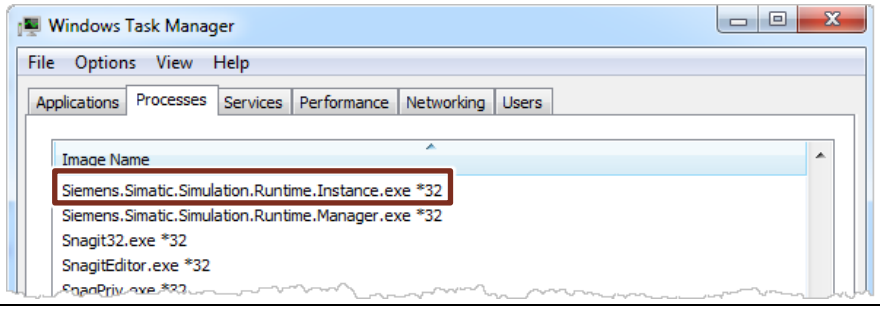
2.3 Bedienung des Anwendungsbeispiels

Hinweis Um über die Anwendung „CoSimulationPlcSimAdv“ die PLCSIM Advanced Simulation zu starten, müssen Sie das Control Panel von PLCSIM Advanced nicht starten.

2.3.1 Virtuellen Controller einschalten

Tabelle 2-20: Virtuellen Controller einschalten

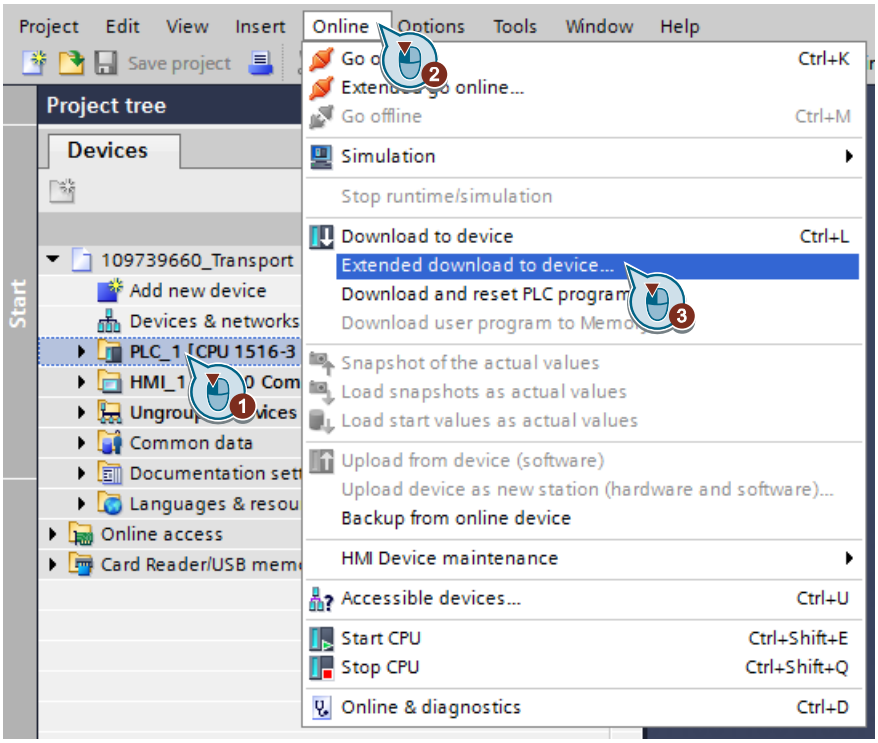
Nr.	Aktion
1.	Entpacken Sie die Datei 109739660_PLCSIM_Advanced_COSIM_APPL_V10.zip“. Öffnen Sie „CoSimulationPlcSimAdv > bin > Release“. Starten Sie die Anwendung „CoSimulationPlcSimAdv.exe“. Es öffnet sich die Bedienoberfläche der Anwendung. In der Statusliste sehen Sie die Information, dass die Instanz „TransportControl“ registriert wurde. 
2.	Optional: Starten Sie den Windows Task Manager und kontrollieren Sie unter „Prozesse“ („Processes“) den gestarteten Prozess des Runtime Managers. 

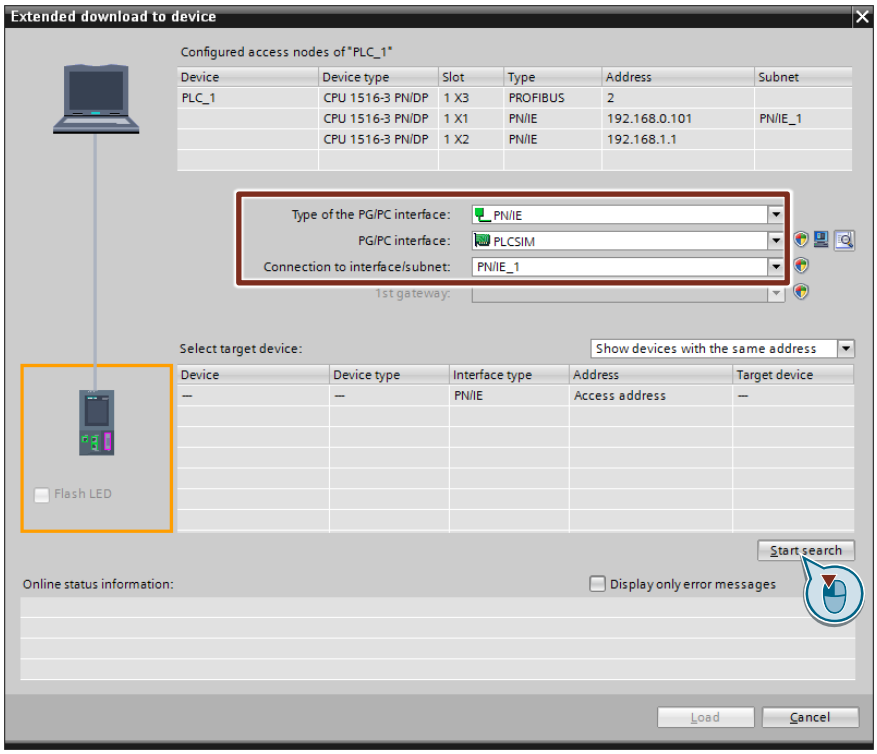
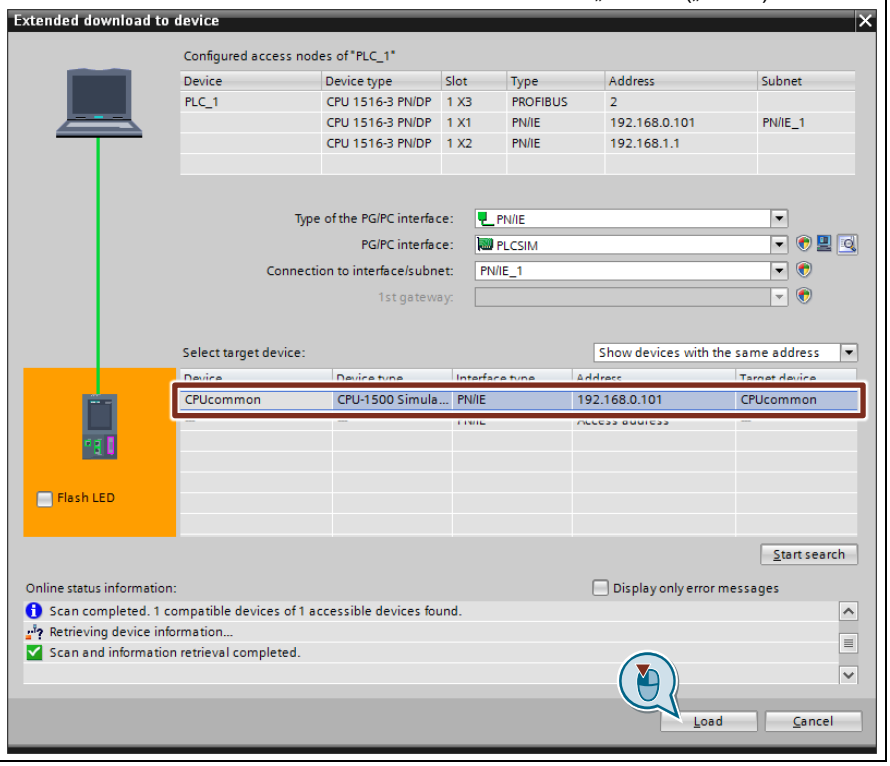
Nr.	Aktion
3.	Klicken Sie in der Bedienoberfläche auf „POWER ON“, um den virtuellen Controller zu starten.  • Die Instanz des virtuellen Controllers wird gestartet. Wenn noch kein Programm in den virtuellen Controller geladen wurde, befindet sich der virtuelle Controller im Zustand „Stop“.  **Hinweis** Wenn Sie die Anwendung zum wiederholten Mal starten und bereits einmal ein Programm in den virtuellen Controller geladen haben, erfolgt ein automatischer Hochlauf aus der Virtual SIMATIC Memory Card und der virtuelle Controller befindet sich im Zustand „Run“.
4.	Optional: Starten Sie den Windows Task Manager und kontrollieren Sie unter „Prozesse“ („Processes“) den gestarteten Prozess der Runtime Instance. 

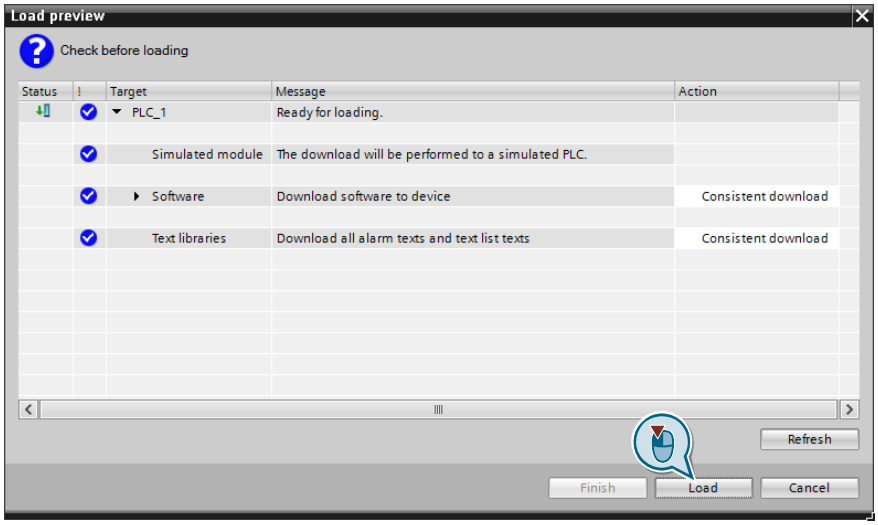
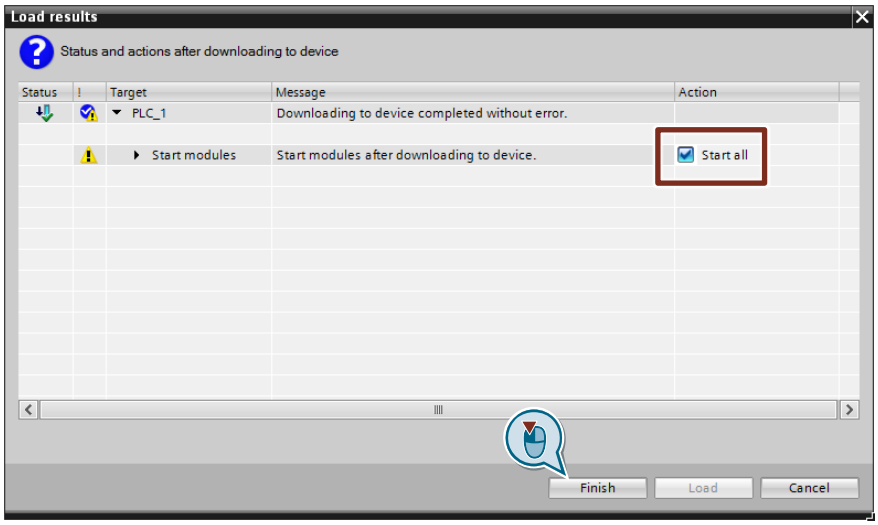
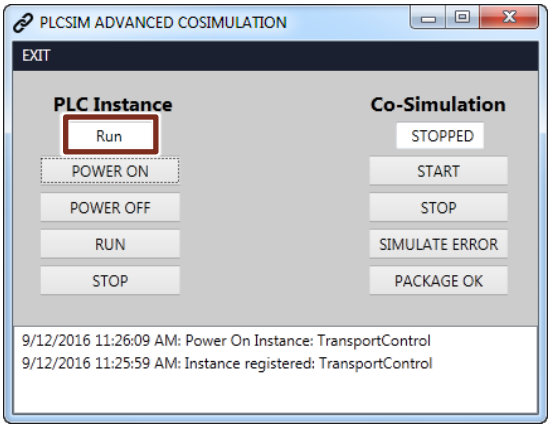
2.3.2 TIA Portal-Projekt in virtuellen Controller laden

Hinweis Um das TIA Portal-Projekt in den virtuellen Controller zu laden, müssen Sie vorher den virtuellen Controller einschalten (siehe Kapitel 2.3.1).

Tabelle 2-21: TIA Portal-Projekt in virtuellen Controller laden

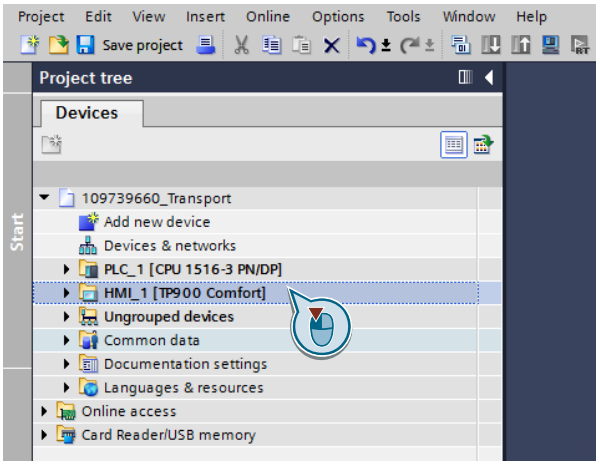
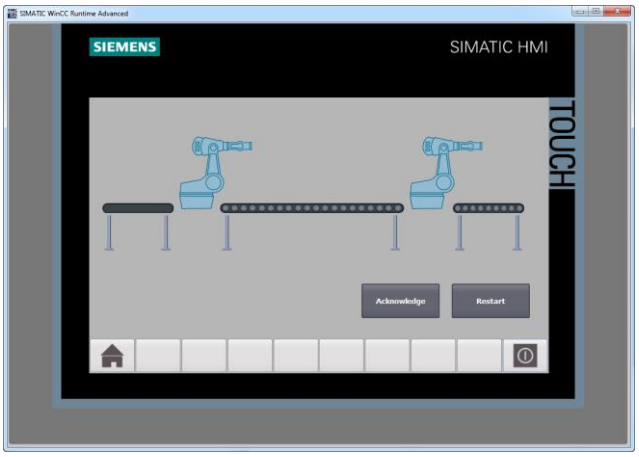
Nr.	Aktion
1.	Öffnen Sie das TIA Portal Projekt „109739660_Transport“ aus dem beigefügten Download „109739660_PLCSIM_Advanced_S7_PROJ_V10.zip“.
2.	1. Wählen Sie in der Projektnavigation die PLC_1 aus. 2. Öffnen Sie das Menü „Online“. 3. Wählen Sie „Erweitertes Laden in Gerät…“ („Extended download to device…“) aus. 

Nr.	Aktion
3.	Stellen Sie als „PG/PC-Schnittstelle“ („PG/PC interface“) „PLCSIM“ ein und klicken Sie auf „Suche starten“ („Start search“). 
4.	Wählen Sie die simulierte CPU aus und klicken Sie auf „Laden“ („Load“). 

Nr.	Aktion
5.	Klicken Sie bei der folgenden Meldung auf „Laden“ („Load“). 
6.	Aktivieren Sie das Optionskästchen „Alle starten“ („Start all“) und klicken Sie auf „Fertig stellen“ („Finish“). 
7.	Kontrollieren Sie nach dem Download, ob der virtuelle Controller in den Zustand „Run“ gewechselt hat. 

2.3.3 WinCC Runtime starten

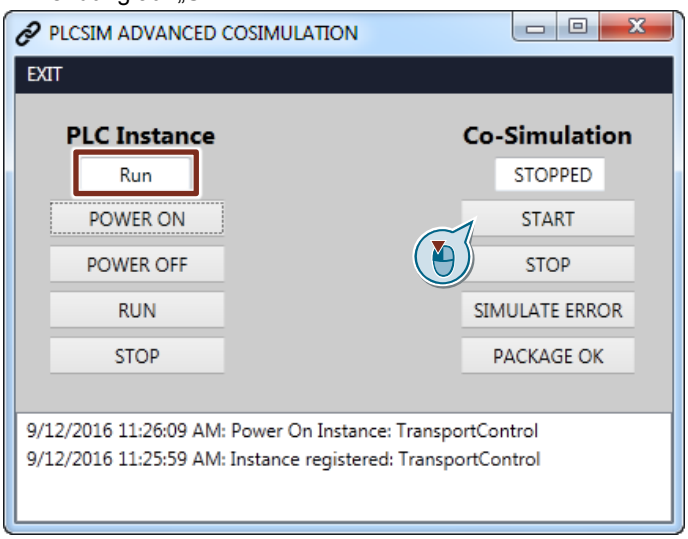
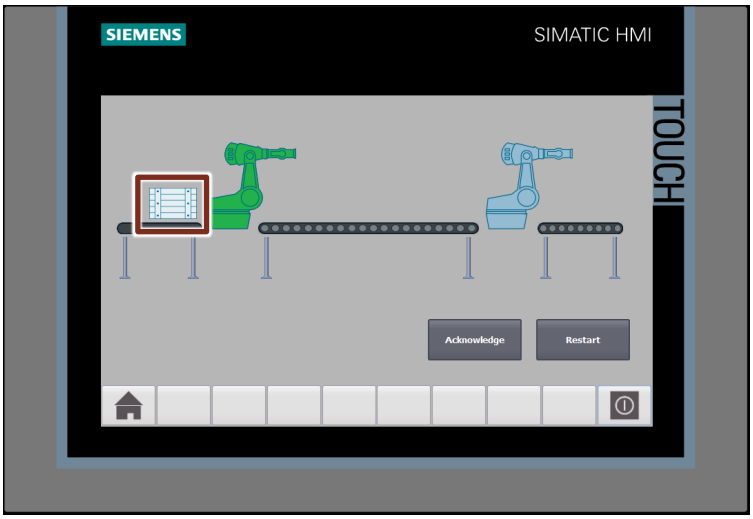
Tabelle 2-22: WinCC Runtime starten

Nr.	Aktion
1.	Wählen Sie in TIA Portal in der Projektnavigation das HMI_1 aus. 
2.	Klicken Sie in der Funktionsleiste des TIA Portals auf „Simulation starten“ („Start simulation“).  • Nach dem Kompilieren startet die WinCC Runtime des SIMATIC HMIs. 

2.3.4 Co-Simulation starten

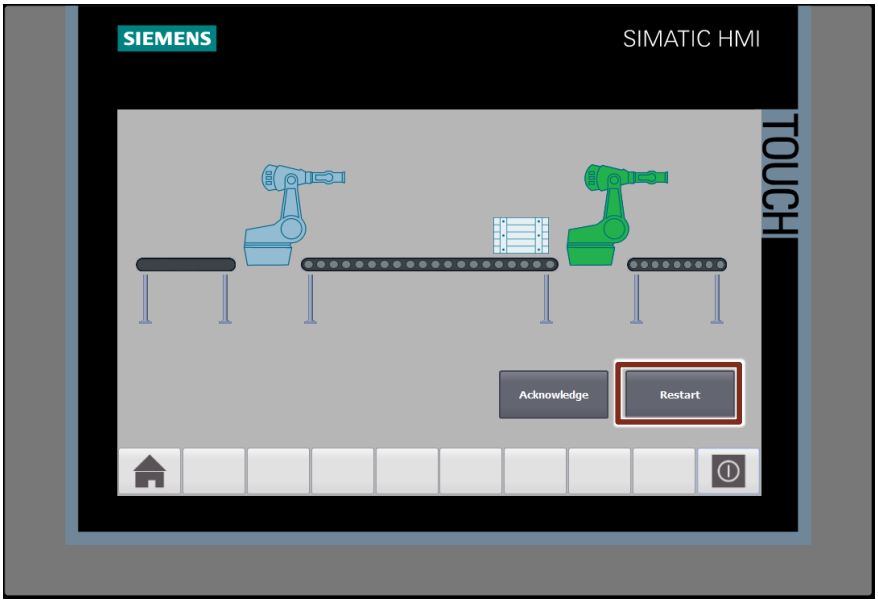
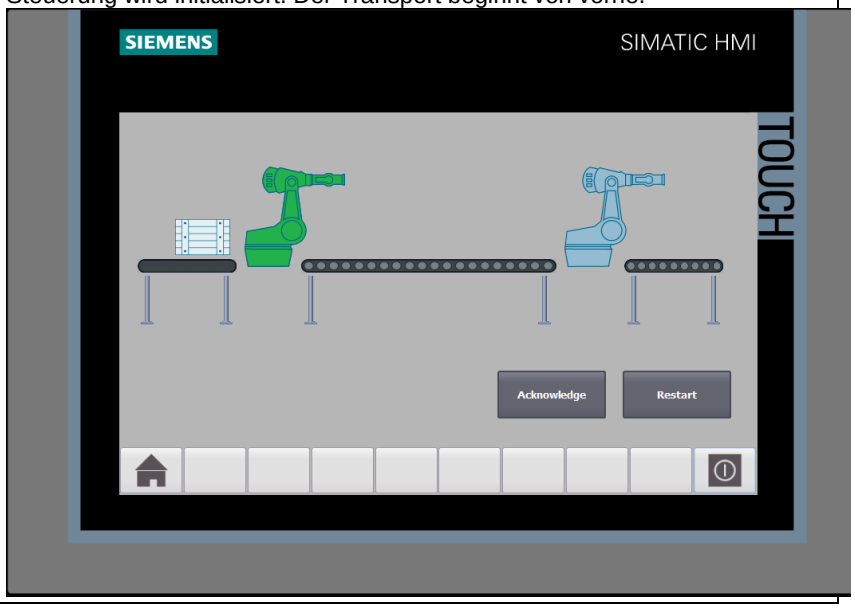
Hinweis Um die Co-Simulation zu bedienen, setzen Sie den virtuellen Controller in den Zustand „Run“.

Tabelle 2-23: Co-Simulation starten

Nr.	Aktion
1.	Um die Co-Simulation zu starten, klicken Sie in der Bedienoberfläche der Anwendung auf „START“.  • Die Co-Simulation platziert eine Kiste an der Startposition und die Transportanlage beginnt mit dem Transport. Nachdem eine Kiste abtransportiert wurde, wird eine neue Kiste an der Startposition platziert. 
2.	Verifizieren Sie in dem simulierten HMI-Bild die Funktion der Transportanlage.

2.3.5 Transportanlage neu starten

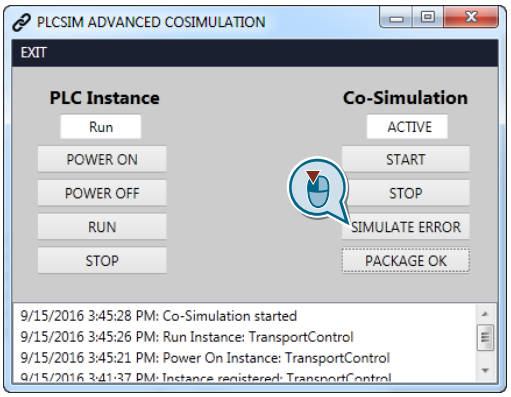
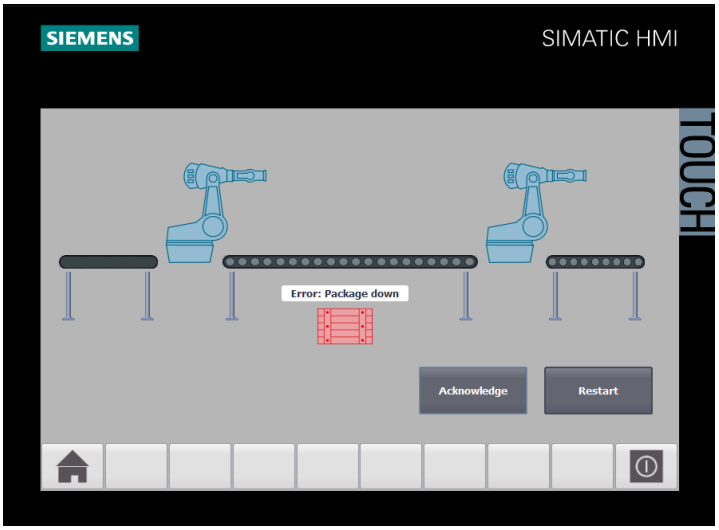
Tabelle 2-24: Anlage neustarten

Nr.	Aktion
1.	Klicken Sie in dem simulierten HMI-Bild auf die Schaltfläche „Restart“.  • Die Co-Simulation platziert die Kiste wieder an der Startposition und die Steuerung wird initialisiert. Der Transport beginnt von vorne. 

2.3.6 Fehler simulieren

Hinweis Um die Co-Simulation zu bedienen, setzen Sie den virtuellen Controller in den Zustand „Run“.

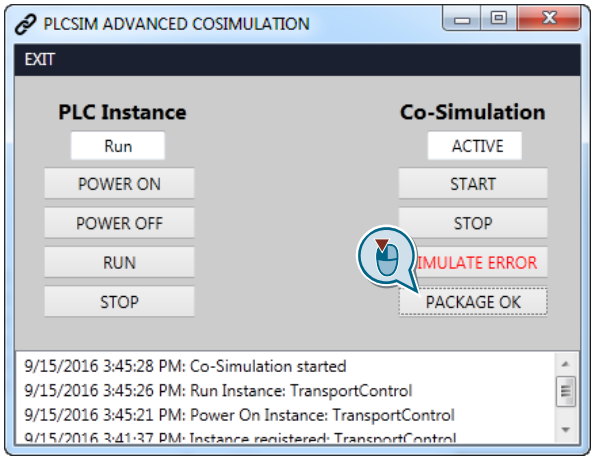
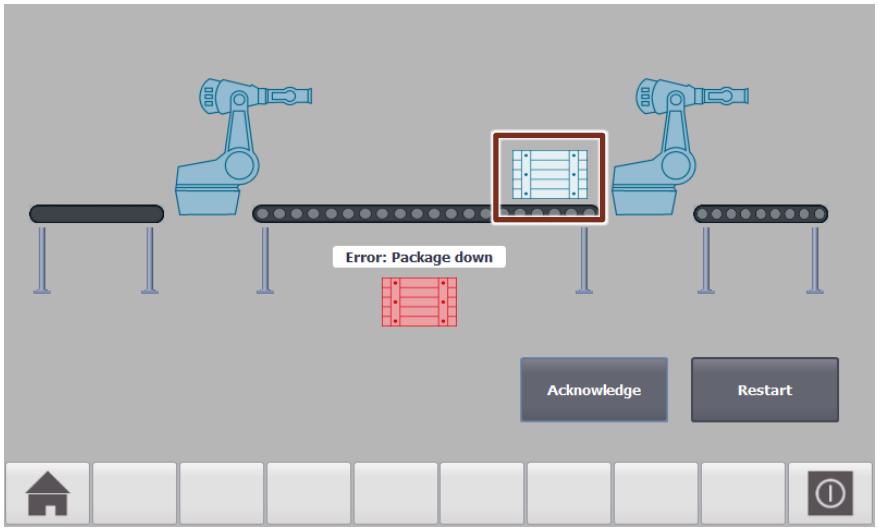
Tabelle 2-25: Fehler simulieren

Nr.	Aktion
1.	Um das Herunterfallen einer Kiste vom Transportband 1 zu simulieren, klicken Sie in der Bedienoberfläche der Anwendung während einer aktiven Co-Simulation auf die Schaltfläche „SIMULATE ERROR“.  • Eine aktivierte Fehlersimulation wird durch eine rote Schriftfarbe angezeigt. 
2.	Verifizieren Sie im simulierten HMI-Bild folgende Zustände: • Die Kiste erreicht ihre Zielposition vor dem zweiten Roboterarm nicht innerhalb der projektierten Überwachungszeit • Der Fehler „Error: Package down“ wird angezeigt. 
3.	Um die Störung zu beheben, haben Sie folgende Optionen: • Transportanlage neu starten (siehe Kapitel 2.3.5) • Fehler beheben und quittieren (siehe Kapitel 2.3.7)

2.3.7 Fehler beheben und quittieren

Hinweis Um die Co-Simulation zu bedienen, setzen Sie den virtuellen Controller in den Zustand „Run“.

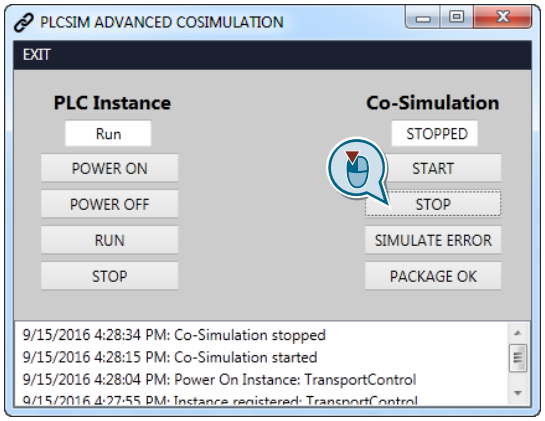
Tabelle 2-26: Fehler beheben und quittieren

Nr.	Aktion
1.	Klicken Sie in der Bedienoberfläche der Anwendung auf die Schaltfläche „PACKAGE OK“. Hierdurch wird das manuelle Aufheben und Auflegen der Kiste auf das Transportband 1 simuliert. 
2.	Verifizieren Sie, dass die Kiste vor dem zweiten Roboterarm platziert ist. 
3.	Klicken Sie in dem simulierten HMI-Bild auf die Schaltfläche „Acknowledge“, um den Fehler zu quittieren und den Transport fortzusetzen. 
4.	Kontrollieren Sie, ob der Transport fortgesetzt wird und die Schrift der Schaltfläche „SIMULATE ERROR“ in der Bedienoberfläche wieder in schwarzer Farbe erscheint. 

2.3.8 Co-Simulation anhalten

Hinweis Um die Co-Simulation zu bedienen, setzen Sie den virtuellen Controller in den Zustand „Run“.

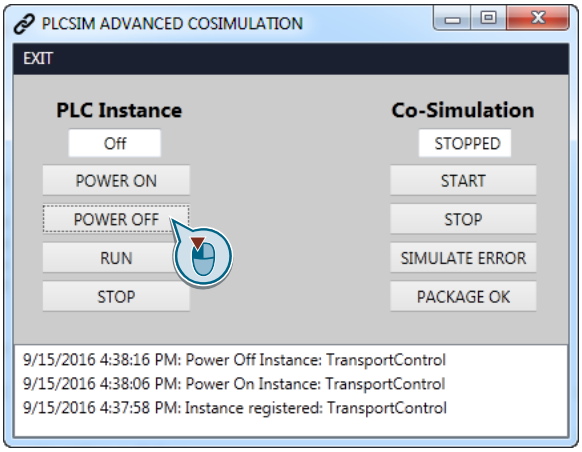
Tabelle 2-27: Co-Simulation anhalten

Nr.	Aktion
1.	Klicken Sie in der Bedienoberfläche der Anwendung auf die Schaltfläche „STOP“.  • Die Co-Simulation wird im aktuellen Schritt angehalten.

Hinweis Wenn Sie die Co-Simulation anhalten während sich die Kiste auf dem Transportband 1 befindet, kann in der Steuerung die Überwachungszeit für das Transportband 1 überschritten werden. Dadurch wird der Fehler einer heruntergefallenen Kiste angezeigt. Starten Sie in diesem Fall die Transportanlage neu (siehe Kapitel 2.3.5), um die Co-Simulation und die Anlagensteuerung wieder zu synchronisieren.

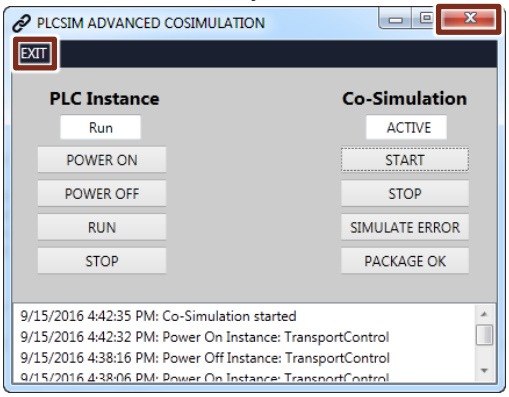
2.3.9 Virtuellen Controller ausschalten

Tabelle 2-28: Virtuellen Controller ausschalten

Nr.	Aktion
1.	Klicken Sie in der Bedienoberfläche der Anwendung auf die Schaltfläche „POWER OFF“. 
2.	Optional: Kontrollieren Sie, ob der virtuelle Controller ausgeschaltet ist und der Prozess der Runtime Instance im Windows Task Manager beendet ist.

2.3.10 Anwendung beenden

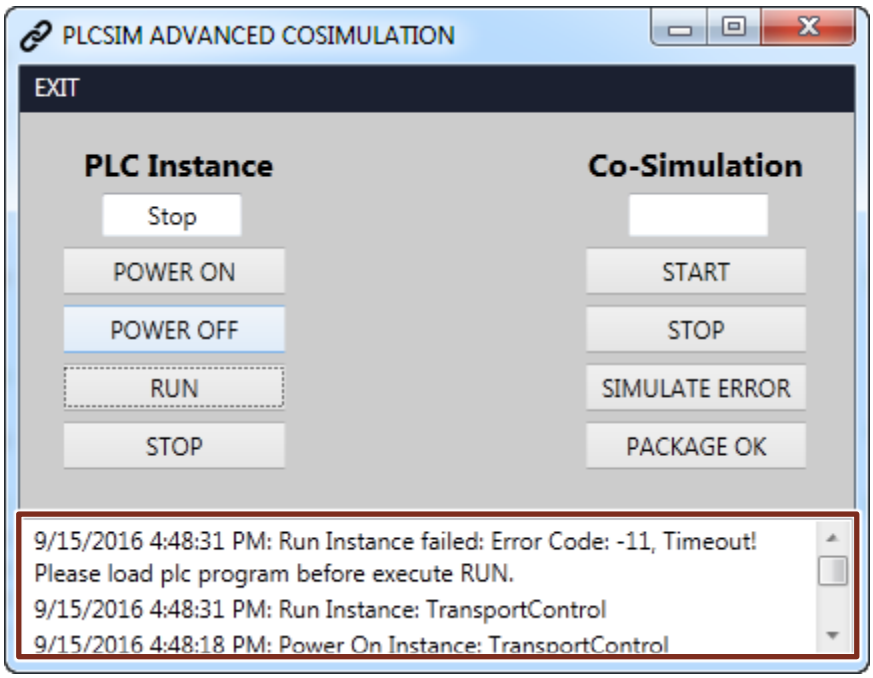
Tabelle 2-29: Anwendung beenden

Nr.	Aktion
1.	Klicken Sie in der Bedienoberfläche der Anwendung auf die Schaltfläche „EXIT“ oder das „Schließen“-Symbol, um die Anwendung zu schließen und zu beenden. 
2.	Optional: Kontrollieren Sie, ob im Windows Task Manager der Prozess des Runtime Managers und der Runtime Instanz beendet ist.

2.3.11 Fehlerhandling

Im Listenfeld der Bedienoberfläche werden die Fehler mit einem Hinweis auf die Ursache angezeigt.

Abbildung 2-2: Listenfeld



3 Wissenswertes

3.1 Grundlagen S7-PLCSIM Advanced

Dieses Kapitel beinhaltet Grundlage zu S7-PLCSIM Advanced. Ausführliche Informationen finden Sie in dem Funktionshandbuch „SIMATIC S7-PLCSIM Advanced“\3\.

https://support.industry.siemens.com/cs/ww/de/view/109739153

3.1.1 Installation

Wenn Sie beabsichtigen mehrere Instanzen von PLCSIM Advanced parallel auszuführen oder die Kommunikation zwischen PLCSIM Advanced V1.0 und Bediengeräten ab Version 14.0 zu simulieren, benötigen Sie eine leistungsfähige Computer-Hardware.

Voraussetzung für die Installation

Bevor Sie die Installation beginnen, müssen Sie folgende Bedingungen erfüllen:

- Hardware und Software des PCs oder des SIMATIC Field PGs erfüllen die Systemanforderungen (siehe Funktionshandbuch „SIMATIC S7-PLCSIM Advanced“, Kapitel „Systemanforderungen“).
- Die Person, welche die Installation durchführt, hat auf dem jeweiligen Computer Administratorrechte.

- Keine anderen Programme sind aktiv. Dies gilt auch für den Siemens Automation License Manager und andere Anwendungen von Siemens.
- Alle S7-PLCSIM-Versionen ab V12 sind deinstalliert.

Hinweis Eine Voraussetzung für die Installation von SIMATIC S7‑PLCSIM Advanced ist, dass sich keine weitere S7‑PLCSIM-Installation auf dem gleichen Rechner befindet.

3.1.2 Unterschied zwischen S7-PLCSIM V14 und PLCSIM Advanced V1.0

Tabelle 3-1: Unterschied zwischen PLCSIM Advanced und PLCSIM V14

Funktion	PLCSIM Advanced V1.0	PLCSIM V14
Runtime	Eigenständig	Zusammen mit STEP 7
Bedienoberfläche	Control Panel	Look&Feel von TIA Portal
Kommunikation	Softbus, TCP/IP	Nur Softbus
Unterstützte CPU-Familien	S7-1500(C,T,F), ET 200SP und ET 200SPF	S7-1200(F), S7-1500(C,T,F), ET 200SP und ET 200SPF
API für Co-Simulation	✓	✗
Webserver	✓	✗
OPC UA	✓	✗
Prozessdiagnose	✓	✓
S7-Kommunikation	✓	Über Softbus
Open User-Kommunikation	✓	Über Softbus
Traces[1]	✓	(✓)
Motion[2]	✓	(✓)
Geschützte Bausteine (KHP)	✓	✗
Multiple Instanzen	Bis zu 16	Bis zu 2
Verteilte Instanzen	✓	✗
Virtuelle Zeit	✓	✗
Anschluss realer CPUs/HMIs	✓	✗
DNS Nutzung	✓	✗
Virtuelle Memory Card	✓	✗

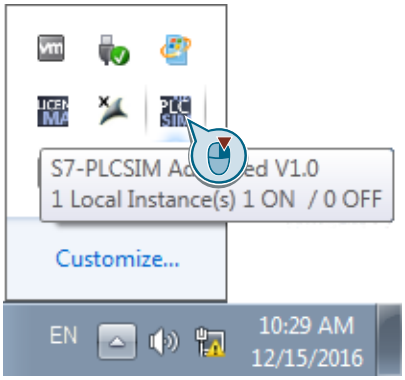
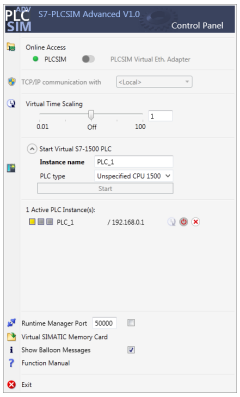
[1] Bei PLCSIM V14 im TIA Portal beobachtbar; bei PLCSIM Advanced V1.0 zusätzlich auch im Webserver beobachtbar.

[2] Bei PLCSIM V14 sind die Achsen immer im Simulationsmodus; bei PLCSIM Advanced V1.0 können die Achsen auch im „Real“ Modus betreiben werden.

3.1.3 Bedienoberfläche

PLCSIM Advanced starten

Tabelle 3-2: PLCSIM Advanced starten

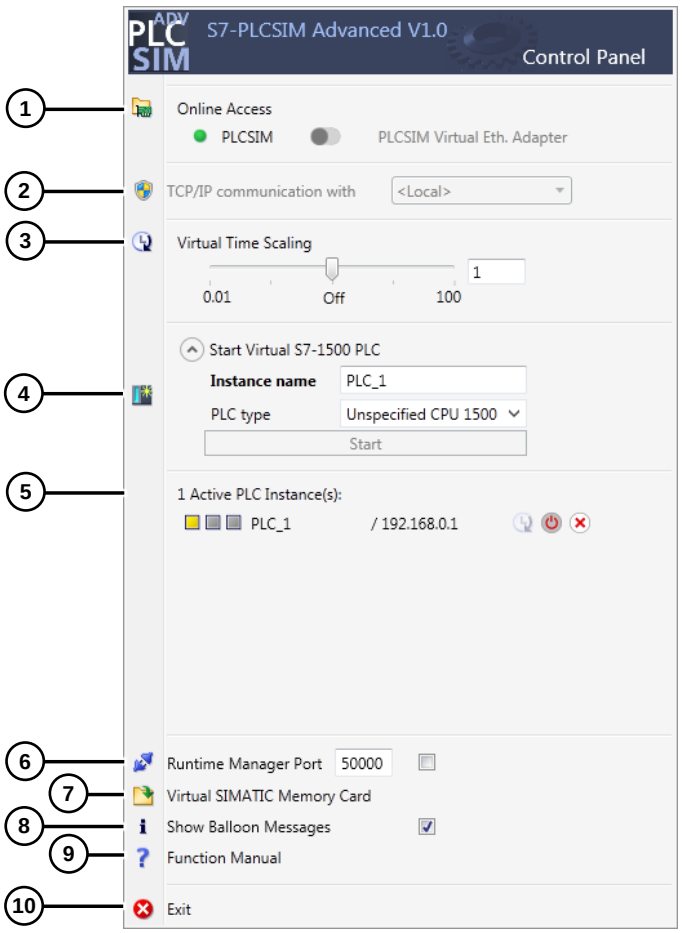
Nr.	Aktion
1.	Doppelklicken Sie auf das PLCSIM Advanced-Symbol auf dem Windows-Desktop, um das Programm zu starten. 
2.	Klicken Sie mit der rechten Maustaste im Infobereich der Taskleiste auf das PLCSIM Advanced-Symbol. **Hinweis:** Über Windows-Funktionen können Sie das Symbol im Infobereich der Taskleiste dauerhaft einblenden.  Die graphische Oberfläche von PLCSIM Advanced wird geöffnet, das Control Panel. Ein Klick auf eine freie Fläche auf dem Desktop schließt die graphische Oberfläche wieder. 

Control Panel

Das Control Panel ist optional und wird nicht zum Betrieb von S7-PLCSIM Advanced über die API benötigt.

3.1 Grundlagen S7-PLCSIM Advanced

Abbildung 3-1: Control Panel



Folgende Tabelle stellt die grundlegenden Funktionen des Control Panels aus der Abbildung 3-1 dar:

Tabelle 3-3: Funktionen des Control Panels

Nr.	Funktion
1.	Schalter zur Auswahl der Kommunikations-Schnittstelle. • **PLCSIM** entspricht der lokalen Kommunikation über Softbus. • **PLCSIM Virtual Ethernet Adapter** entspricht der Kommunikation über TCP/IP.
2.	Auswahl der Netzwerkkarte für die verteilte Kommunikation.
3.	Mit dem Schieberegler wählen Sie den Skalierungsfaktor für die virtuelle Zeit.
4.	Unter „**Instance name**“ geben Sie einen eindeutigen Namen (mind. drei Zeichen) für die Instanz ein. Wenn der Name bereits im Verzeichnis der Virtual SIMATIC Memory Card existiert, dann wird die bereits vorhandene Instanz gestartet. Unter „**PLC-Type**“ selektieren Sie den zu simulierenden CPU-Typ. Zusätzlich werden hier die Eingabefelder **IP-Adresse**, **Subnetz Maske** und **Default Gateway** sichtbar, wenn Sie die Kommunikations-Schnittstelle auf „PLCSIM Virtual Ethernet Adapter“ umschalten.
5.	Die Liste zeigt die **lokal** vorhandenen Instanzen. Die Bedeutung der LED und der Schaltflächen wird angezeigt, wenn Sie den Mauszeiger darüber bewegen.
6.	Hier öffnen Sie einen Port auf dem lokalen PC. Über den eingestellten Port kann eine Remote-Verbindung zu einem weiteren Runtime Manager hergestellt werden.

Nr.	Funktion
7.	Hier öffnen Sie ein Explorer-Fenster mit dem Pfad zur virtuellen Speicherkarte.
8.	Hier deaktivieren Sie die PLCSIM Advanced-Meldungen in der Windows-Taskleiste für die Dauer der Bedienung.
9.	Hier öffnen Sie das Funktionshandbuch „SIMATIC S7-PLCSIM Advanced“ im Standard PDF-Viewer.
10.	Schaltet alle lokalen Instanzen ab und meldet Sie vom Runtime Manager ab.

3.1.4 Generelle Eigenschaften

Anwendungsgebiete

- Funktionstest des STEP 7-Programms – auch im Kontext einer Anlage/Maschine
- Virtuelle Funktionsvalidierung bis hin zur virtuellen Inbetriebnahme
- Ausbildung und Training für S7-1500 Automation ohne Hardware (Operator Training)
- Factory Acceptance Test (FAT)

Vorteile

Der Einsatz von S7-PLCSIM Advanced bietet zahlreiche Vorteile:

- Frühe Fehlererkennung und Validierung der Funktionalität → Hohe Qualität des STEP 7 Programm-Codes
- Keine reale S7-CPU-Hardware notwendig → Einsparung von Hardware-Kosten
- Effizienzsteigerung durch Optimierung von Programmteilen
- Verkürzung von Entwicklungszeiten bzw. des Time-to-Market
- Risiken für die Inbetriebnahme reduzieren
- Frühes Training der Bediener möglich (Operator Training)

Lokale und verteilte Kommunikation

Neben der lokalen Kommunikation über Softbus bietet S7-PLCSIM Advanced einen vollwertigen Ethernet-Anschluss und kann somit auch verteilt kommunizieren.

Für die Kommunikation zwischen STEP 7 und den Instanzen der PLCSIM Advanced stehen folgende Kommunikationswege und -möglichkeiten zur Verfügung:

Tabelle 3-4: Kommunikationswege

Kommunikationswege	Lokal		Verteilt
Protokoll	Softbus	TCP/IP	TCP/IP
Kommunikationsschnittstelle in PLCSIM Advanced	PLCSIM	PLCSIM Virtual Ethernet Adapter	PLCSIM Virtual Ethernet Adapter
STEP 7 und Instanzen	auf einem PC / einer VM	auf einem PC / einer VM	verteilt

Tabelle 3-5: Kommunikationsmöglichkeiten

Kommunikationsmöglichkeiten	Lokal		Verteilt
Protokoll	Softbus	TCP/IP	TCP/IP
zwischen STEP 7 und Instanzen	ja	ja	ja
zwischen Instanzen untereinander	ja	ja	ja
über OPC UA und Webserver	nein	ja	ja
zwischen einer Instanz und einer realen Hardware-CPU	nein	nein	ja
zwischen einer Instanz und einer realen HMI V14	nein	nein	ja
zwischen einer Instanz und einer simulierten HMI V14	ja	ja	nein

Verteilte Kommunikation

Verteilte Kommunikation bedeutet, dass S7-PLCSIM Advanced Instanzen über den PLCSIM Virtual Switch an ein Netzwerk angebunden werden können. Somit können Instanzen auch rechnerübergreifend mit anderen Geräten kommunizieren.

Eine Kommunikation ist mit realen oder simulierten CPUs bzw. HMIs möglich.

Virtuelles Zeitverhalten

Für die Simulation nutzt der virtuelle Controller intern zwei Arten von Uhren:

- **Virtuelle Uhr**

  Bildet die Basis für das Anwenderprogramm und wird von Komponenten genutzt, die von Steuerungsprozessen abhängig sind (z. B. zyklische OBs, Zyklusüberwachung, minimale Zykluszeit, Systemzeit und Zeitberechnungen).

  Die virtuelle Uhr kann beschleunigt oder verlangsamt werden.

- **Reale Uhr**

  Kann nicht beschleunigt oder verlangsamt werden und wird von Komponenten genutzt, die nicht von Steuerungsprozessen abhängig sind, z. B. die Kommunikation mit STEP 7.

Mit PLCSIM Advanced können Sie zu Testzwecken die virtuelle Uhr verlangsamen oder beschleunigen.

Hinweis Wenn Sie während der Simulation eines Programms einen Zykluszeit-Fehler erhalten, verlangsamen Sie die virtuelle Uhr, indem Sie den Skalierungsfaktor kleiner 1 wählen.

3.1.5 Simulation

Unterstützte CPUs

S7-PLCSIM Advanced V1.0 unterstützt die Simulation aller Hardware CPUs der S7-1500-Produktfamilie. Es werden alle Firmware-Versionsstände unterstützt, wobei V2.0 empfohlen wird.

Tabelle 3-6: Unterstützte Hardware

Typ		Artikelnummer
Standard-CPUs	CPU 1511-1 PN CPU 1513-1 PN CPU 1515-2 PN CPU 1516-3 PN/DP CPU 1517-3 PN/DP[2] CPU 1518-4 PN/DP[2] CPU 1518-4 PN/DP ODK[2, 3]	6ES7511-1AK01-0AB0 6ES7513-1AL01-0AB0 6ES7515-2AM01-0AB0 6ES7516-3AN01-0AB0 6ES7517-3AP00-0AB0 6ES7518-4AP00-0AB0 6ES7518-4AP00-3AB0
Fehlersichere CPUs	CPU 1511F-1 PN CPU 1513F-1 PN CPU 1515F-2 PN CPU 1516F-3 PN/DP CPU 1517F-3 PN/DP[2] CPU 1518F-4 PN/DP[2] CPU 1518F-4 PN/DP ODK[2, 3]	6ES7511-1FK01-0AB0 6ES7513-1FL01-0AB0 6ES7515-2FM01-0AB0 6ES7516-3FN01-0AB0 6ES7517-3FP00-0AB0 6ES7518-4FP00-0AB0 6ES7518-4FP00-3AB0
Kompakt-CPUs[1]	CPU 1511C-1 PN CPU 1512C-1 PN	6ES7511-1CK00-0AB0 6ES7512-1CK00-0AB0
ET 200SP-CPUs	CPU 1510SP-1 PN CPU 1510SP F-1 PN CPU 1512SP-1 PN CPU 1512SP F-1 PN	6ES7510-1DJ01-0AB0 6ES7510-1SJ01-0AB0 6ES7512-1DK01-0AB0 6ES7512-1SK01-0AB0
Technologie-CPUs	CPU 1511T-1 PN CPU 1515T-2 PN CPU 1517T-3 PN/DP[2, 4] CPU 1517TF-3 PN/DP[2, 4]	6ES7511-1TK01-0AB0 6ES7515-2TM01-0AB0 6ES7517-3TP00-0AB0 6ES7517-3UP00-0AB0

[1] Es wird nicht die Onboard-Peripherie innerhalb der Kompakt-CPUs simuliert. Die Simulations-Schnittstelle entspricht dem Prozessabbild.

[2] Es wird nicht die ODK-Funktionalität dieser CPU simuliert.

[3] Die Simulation dieser CPU unterstützt nur 5120 Motion Control-Ressourcen.

[4] Die Simulation dieser CPU unterstützt nur 64 Kurvenscheiben.

Simulierbare Kommunikationsdienste

S7-PLCSIM Advanced V1.0 unterstützt die Simulation folgender Kommunikationsdienste:

Tabelle 3-7: Simulierbare Kommunikationsdienste

Kommunikationsdienste	Beschreibung
PG-Kommunikation	Zur Inbetriebnahme, Test und Diagnose
Offene Kommunikation über TCP/IP	TSEND_C / TRCV_C TSEND / TRCV TCON T_DISCON
Offene Kommunikation über ISO-on-TCP	TSEND_C / TRCV_C TSEND / TRCV TCON T_DISCON
Offene Kommunikation über UDP[1]	TUSEND / TURCV TCON T_DISCON
Kommunikation über Modbus TCP	MB_CLIENT MB_SERVER
E-Mail[1]	TMAIL_C
S7-Kommunikation	PUT / GET BSEND / BRCV USEND / URCV
OPC UA Server[1]	Datenaustausch mit OPC UA Clients
Webserver[1]	Datenaustausch über HTTP

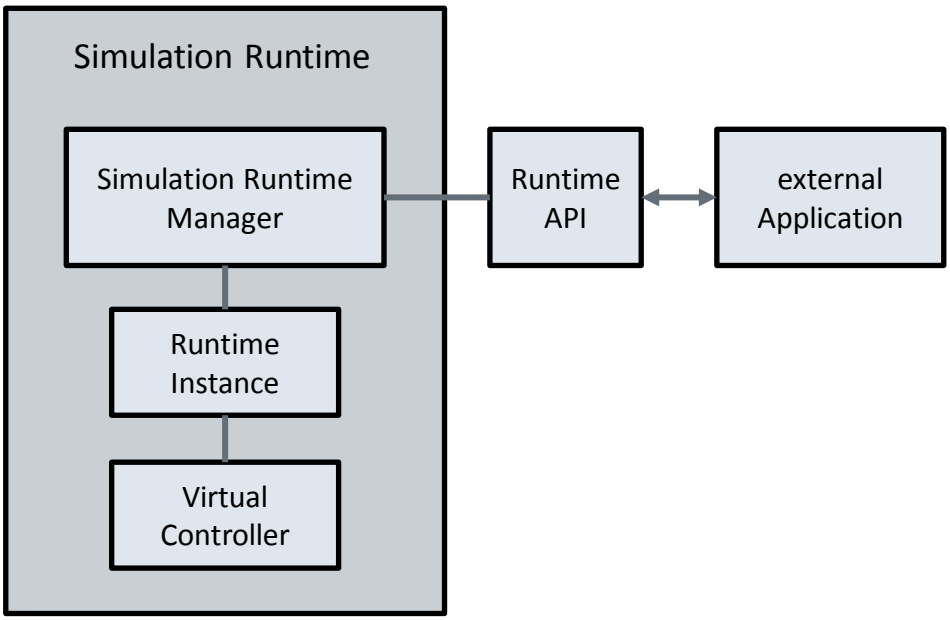
[1)] Nur über die Kommunikations-Schnittstelle „PLCSIM Virtual Ethernet Adapter“ (TCP/IP).

3.1.6 Anwenderschnittstellen (API)

PLCSIM Advanced ermöglicht über die Anwenderschnittstellen die Interaktion mit externen Applikationen, z. B. eigene C++/C#-Programme oder Software zur Simulation von Produktionsmaschinen und Anlagen.

Über die Runtime API greifen Sie auf die Simulation Runtime zu. Der Simulation Runtime Manager verwaltet die Runtime Instanzen. Die Runtime Instanzen laden die Bibliotheken des virtuellen Controllers.

Abbildung 3-2: Externe Applikation und Simulation Runtime



Komponenten der Simulation Runtime

Folgende Tabelle stellt die relevanten Komponenten für den Umgang mit der Simulation Runtime der PLCSIM Advanced dar.

Tabelle 3-8: Simulation Runtime Komponenten

Komponente	Beschreibung
Siemens.Simatic.Simulation.Runtime.Manager.exe	Ein Windows-Prozess, der im Hintergrund abläuft. Hauptkomponente der Runtime, die alle weiteren Runtime Komponenten verwaltet. Der Prozess wird automatisch gestartet, sobald eine Applikation versucht die Runtime API zu initialisieren. Der Prozess wird automatisch beendet, sobald keine Applikation mehr läuft, welche die Runtime API initialisiert hat.
Siemens.Simatic.Simulation.Runtime.Instance.exe	Der Prozess der Instanz, der eine DLL eines virtuellen Controllers lädt. Jeder virtuelle Controller erzeugt jeweils seinen eigenen Prozess.
Siemens.Simatic.Simulation.Runtime.Api.x86.dll Siemens.Simatic.Simulation.Runtime.Api.x64.dll	API-Bibliotheken, die eine Anwendung laden muss, um die Simulation Runtime zu verwenden. Die Bibliotheken enthalten Schnittstellen für Native und Managed Code. Die "Runtime.Api.x86.dll" wird ausschließlich von 32 Bit-Anwendungen geladen, die Runtime.Api.x64.dll von 64 Bit-Anwendungen. Die DLLs basieren auf .NET Framework 4.0.

Komponente	Beschreibung
SimulationRuntimeApi.h	Headerdatei, die alle Datentypen beschreibt, die eine Native C++ Anwendung benötigt, um die API-Bibliothek zu verwenden.

Runtime API

Die Runtime API stellt einer externen Applikation die Schnittstelle zur Verfügung. Diese enthalten Funktionen, mit denen Sie z. B. Instanzen erzeugen, den Betriebszustand eines virtuellen Controllers ändern und I/O-Daten austauschen.

Folgende Schnittstellen werden in diesem Anwendungsbeispiel verwendet:

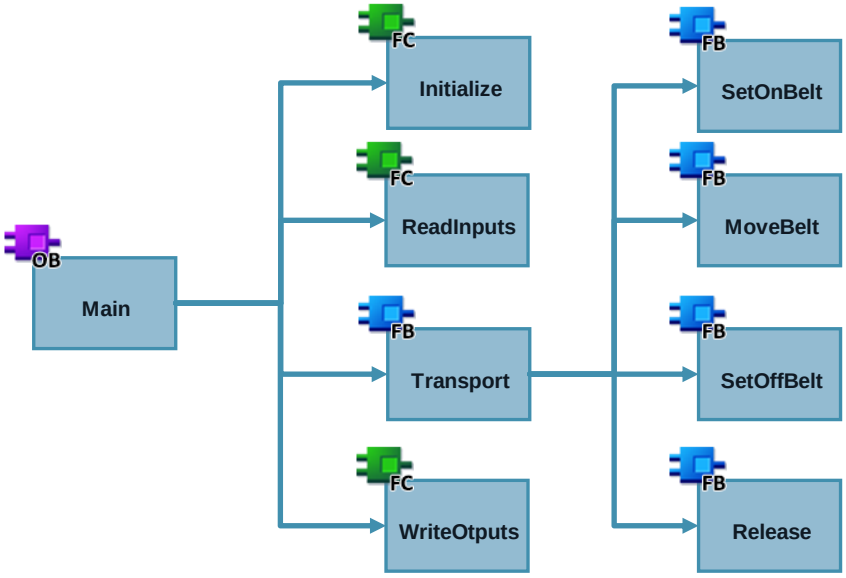
- **ISimulationRuntimeManager**
  Schnittstelle des Runtime Managers. Diese Schnittstelle wird benutzt, um neue Runtime Instanzen zu registrieren, bereits bestehende zu durchsuchen und eine Schnittstelle einer registrierten Instanz zu erhalten. In einem Runtime Manager können bis zu 16 Instanzen registriert werden.
- **IInstance**
  Schnittstelle einer Runtime Instanz. Diese Schnittstelle wird benutzt, um den Betriebszustand eines virtuellen Controllers zu ändern und I/O-Daten auszutauschen. Jede Instanz hat einen eindeutigen Namen und eine ID.

3.2 Details zur Funktionsweise

3.2.1 TIA Portal-Programm

Aufbau

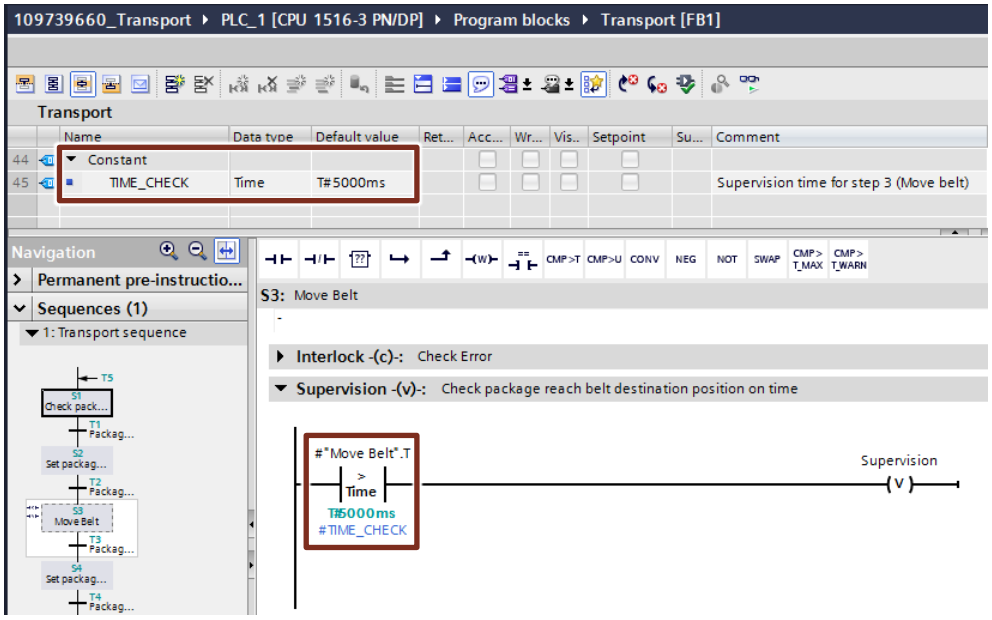
Abbildung 3-3: Aufbau des TIA Portal-Programms



Überwachungszeit des Transportband 1

Die Überwachungszeit für das Transportband 1 ist im Funktionsbaustein „Transport“ als eine Zeitkonstante mit dem Wert 5000 ms definiert. Im Schritt 3 „Move belt“ wird die gesamte Aktivierungszeit des Schrittes überwacht.

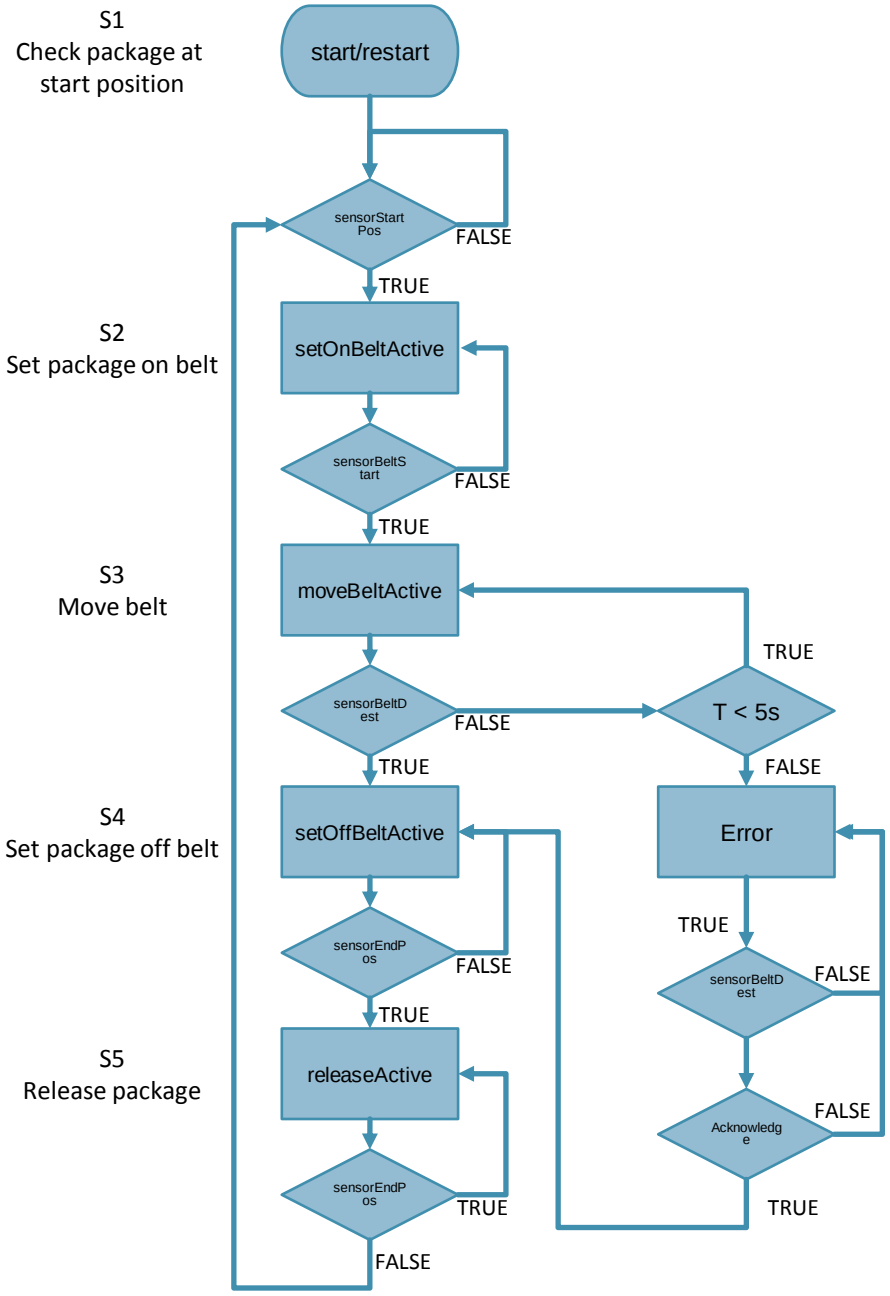
Abbildung 3-4: Überwachungszeit



Ablaufsteuerung

Die komplette Ablaufsteuerung der Transportanlage ist im Funktionsbaustein „Transport“ mit S7-GRAPH programmiert. Folgendes Ablaufdiagramm stellt den Ablauf der Sequenz dar.

Abbildung 3-5: Ablaufsteuerung der Transportanlage



3.2.2 Co-Simulations-Programm

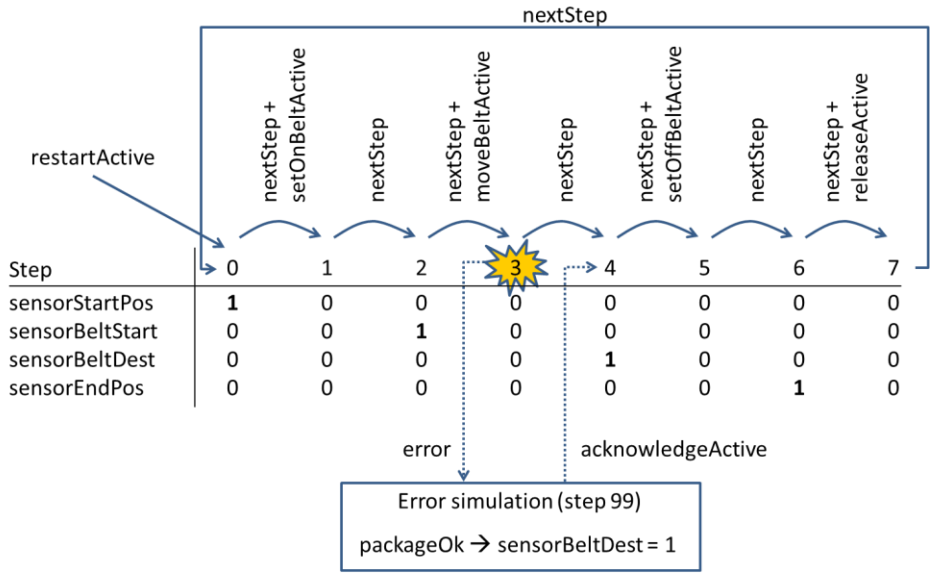
Übersicht

Das Co-Simulations-Programm ist im Visual Studio-Projekt in der Methode „CoSimProgramm()“ der Klasse „Cosimulation.cs“ programmiert.

Das Programm ist in einer Schrittkette (Switch-Case) programmiert. Für die Simulation eines Sensors und den Betrieb einer Maschine wird in dem entsprechenden Schritt ein Timer gestartet, nach dessen Ablauf die Variable „nextStep“ gesetzt wird.

Folgende Abbildung stellt den Programmablauf der Co-Simulation dar.

Abbildung 3-6: Co-Simulation Programmablauf



Fehlersimulation

Durch das Setzen der Variablen „error“ wird simuliert, dass eine Kiste vom Transportband 1 fällt.

Wenn die Variable „error“ gesetzt ist, unterbricht das Programm in Schritt 3 den regulären Ablauf und springt zum Schritt 99 (Fehlersimulation).

Die Fehlersimulation wird beendet, wenn die Kiste wieder auf das Transportband 1 gesetzt („packageOk“) und in der Steuerung der Fehler quittiert wird (acknowledgeActive) oder die Anlage neu gestartet wird (restartActive).

Hinweis Weitere Beschreibung des Co-Simulation-Programms finden Sie in den Kommentaren im Programmcode.

4 Anhang

4.1 Service und Support

Industry Online Support

Sie haben Fragen oder brauchen Unterstützung?

Über den Industry Online Support greifen Sie rund um die Uhr auf das gesamte Service und Support Know-how sowie auf unsere Dienstleistungen zu.

Der Industry Online Support ist die zentrale Adresse für Informationen zu unseren Produkten, Lösungen und Services.

Produktinformationen, Handbücher, Downloads, FAQs und Anwendungsbeispiele – alle Informationen sind mit wenigen Mausklicks erreichbar:
https://support.industry.siemens.com/ .

Technical Support

Der Technical Support von Siemens Industry unterstützt Sie schnell und kompetent bei allen technischen Anfragen mit einer Vielzahl maßgeschneiderter Angebote – von der Basisunterstützung bis hin zu individuellen Supportverträgen.

Anfragen an den Technical Support stellen Sie per Web-Formular:
www.siemens.de/industry/supportrequest .

Serviceangebot

Unser Serviceangebot umfasst u. a. folgende Services:

- Produkttrainings
- Plant Data Services
- Ersatzteilservices
- Reparaturservices
- Vor-Ort und Instandhaltungsservices
- Retrofit- und Modernisierungsservices
- Serviceprogramme und Verträge

Ausführliche Informationen zu unserem Serviceangebot finden Sie im Servicekatalog:
https://support.industry.siemens.com/cs/sc

Industry Online Support App

Mit der App "Siemens Industry Online Support" erhalten Sie auch unterwegs die optimale Unterstützung. Die App ist für Apple iOS, Android und Windows Phone verfügbar.
https://support.industry.siemens.com/cs/de/de/sc/2067

4.2 Links und Literatur

Tabelle 4-1

Nr.	Thema
\1\	Industry Online Support https://support.industry.siemens.com
\2\	SIMATIC S7 PLCSIM Advanced: Co Simulation via API https://support.industry.siemens.com/cs/ww/de/view/109739660
\3\	SIMATIC S7-PLCSIM Advanced Funktionshandbuch https://support.industry.siemens.com/cs/ww/de/view/109739153

4.3 Änderungsdokumentation

Tabelle 4-2

Version	Datum	Änderung
V1.0	12/2016	Erste Ausgabe